MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

Automated Circuit Extraction from Mask Descriptions of MOS Networks

Steven Paul McCormick

AD-A151 208

Technical Report 503

February 1984

DTIC
SELECTE
MAR 1 3 1985
S E D

Massachusetts Institute of Technology
Research Laboratory of Electronics
Cambridge, Massachusetts 02139

85   017

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	

4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 8 5 - 0 1 7 4

6a. NAME OF PERFORMING ORGANIZATION Massachusetts Institute of Technology Research Laboratory of Electronics	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AFOSR /NE
6c. ADDRESS (City, State and ZIP Code) 77 Massachusetts Avenue Cambridge, MA 02139		7b. ADDRESS (City, State and ZIP Code) Bldg 410 BAFB DC 20332-6448

8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Office of Scientific Research	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER Contract F49620-81-C-0054
8c. ADDRESS (City, State and ZIP Code) Bolling Air Force Base, D. C. 20332		10. SOURCE OF FUNDING NOS.

10. SOURCE OF FUNDING NOS.			
PROGRAM ELEMENT NO. 61102F	PROJECT NO. 2305/B3	TASK NO.	WORK UNIT NO.

11. TITLE (Include Security Classification)
Automated Circuit Extraction from Mask

12. PERSONAL AUTHOR(S) Descriptions of MOS Networks   by   S. P. McCormick

13a. TYPE OF REPORT Technical Report 503	13b. TIME COVERED FROM 15 mar 81 TO 29 Sep 83	14. DATE OF REPORT (Yr., Mo., Day) February 1984	15. PAGE COUNT 123

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Automated circuit extration Integrated circuit Geometric mask information

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

An automated circuit extractor generates an equivalent circuit description of an integrated circuit (IC) entirely from the geometric mask information. By analyzing the circuit description, IC performance can be estimated without having the IC design implemented. This thesis presents a methodology for accurate extraction of interconnection resistance, inter-nodal capacitance, ground capacitance, and transistor dimensions—circuit parameters important in characterizing the speed, noise-immunity, and static performance of designs for modern MOS technologies. Extracting each circuit parameter follows a general, numerical extraction algorithm with high accuracy. However, where possible, the general algorithms are replaced with simple techniques that do not sacrifice accuracy but execute much faster. Vital to the extraction methodology is a geometric operation that decomposes regions into domains appropriate for specialized algorithms and general algorithms.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED X SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Robert W Carter, LT Col USAF	22b. TELEPHONE NUMBER (Include Area Code) (202) 767-4931	22c. OFFICE SYMBOL NE

DD FORM 1473, 83 APR   EDITION OF 1 JAN 73 IS OBSOLETE.

Automated Circuit Extraction from Mask Descriptions of MOS Networks

Steven Paul McCormick

Technical Report 503

February 1984

Massachusetts Institute of Technology

Research Laboratory of Electronics

Cambridge, Massachusetts 02139

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC

NOTICE OF

This tech

appro 1 -12.

Dist

MATTHE

Chief, Technical Information Division

Automated Circuit Extraction from Mask Descriptions of MOS Networks

by

Steven Paul McCormick

Submitted to the
Department of Electrical Engineering and Computer Science
on February 17, 1984 in partial fulfillment of the requirements
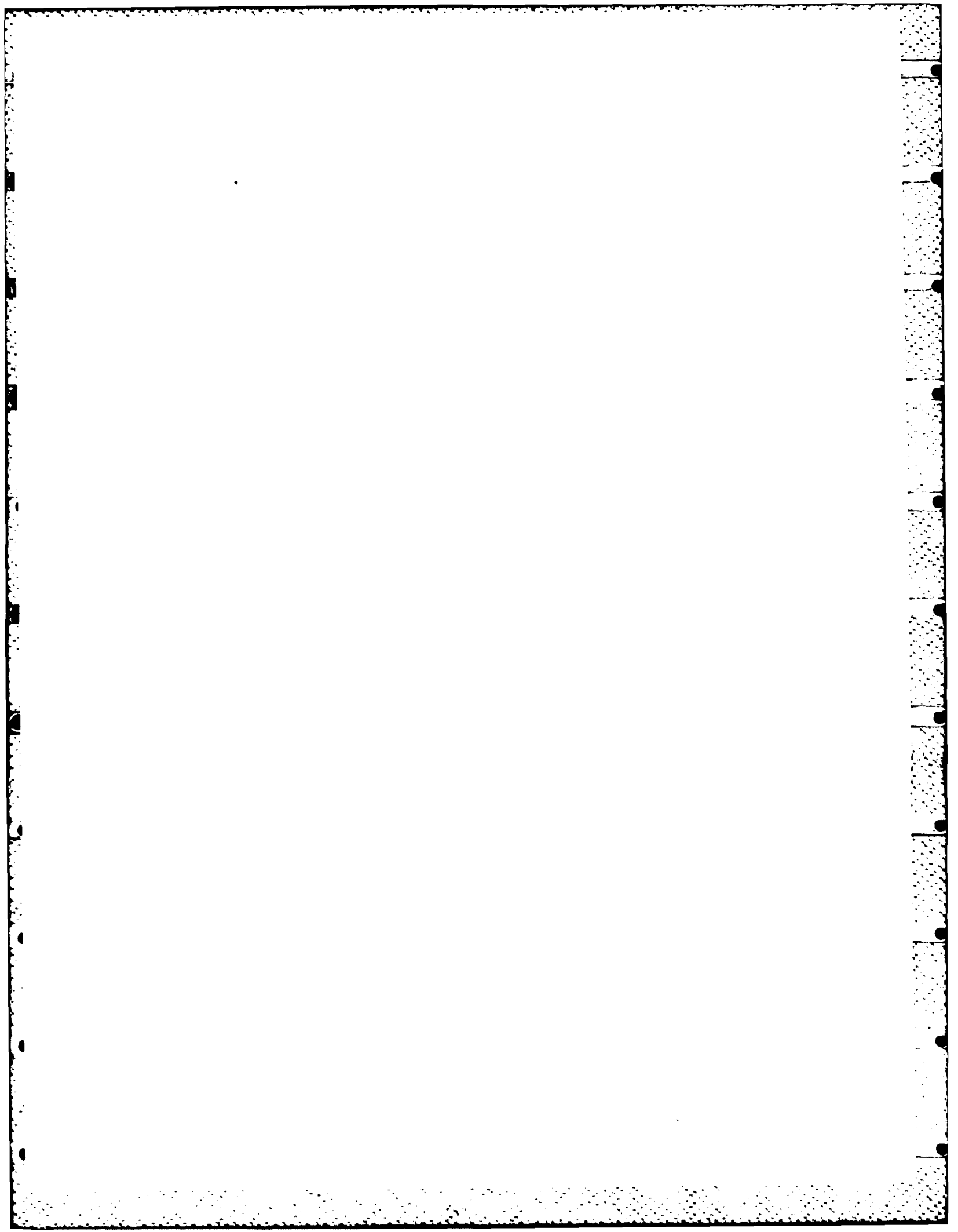for the Degree of Master of Science.

Abstract

An automated circuit extractor generates an equivalent circuit description of an integrated circuit (IC) entirely from the geometric mask information. By analyzing the circuit description, IC performance can be estimated without having the IC design implemented. This thesis presents a methodology for accurate extraction of interconnection resistance, inter-nodal capacitance, ground capacitance, and transistor dimensions—circuit parameters important in characterizing the speed, noise-immunity, and static performance of designs for modern MOS technologies. Extracting each circuit parameter follows a general, numerical extraction algorithm with high accuracy. However, where possible, the general algorithms are replaced with simple techniques that do not sacrifice accuracy but execute much faster. Vital to the extraction methodology is a geometric operation that decomposes regions into domains appropriate for specialized algorithms and general algorithms.

Thesis Supervisor: Jonathan Allen

Title: Professor of Electrical Engineering and Computer Science

2

Acknowledgments

I thank Jonathan Allen for the initial project suggestion, for his continued support and valuable recommendations and ideas on extraction.

I thank the users of EXCL who have tested the extraction methodologies, uncovered problems, and made suggestions for improvement. In particular, I thank Donald Baltus who made the first exhaustive test of EXCL.

I thank William Evans for his valuable discussions on extraction and future needs of MOS IC design.

Finally, I thank Robert Armstrong for HPDRAW—the system that generated the figures in this thesis.

Table of Contents

Table of Figures

Table of Tables

CHAPTER ONE

Introduction

An integrated circuit designer generally wishes to locate and fix circuit design problems before dedicating his design to the expensive IC fabrication line. Inadequate speed, degraded logic levels, and excessive noise are just some of the circuit problems that plague the IC designer. Not too many years ago, extracting active device and connectivity information was sufficient to characterize the IC's circuit behavior. However, as IC structures are scaled down in size, problems associated with IC interconnections become particularly acute, for the effects from parasitic resistances and capacitances begin to dominate over device effects. Today, a necessary step in locating potential circuit problems is extracting equivalent circuits for active devices and interconnections.

Previous computer programs have been written [1, 2, 3] for extracting transistor information and interconnection capacitance formed by overlapping layers. These extractors are proficient in verifying logical correctness, but not circuit performance correctness. None extract arbitrary inter-nodal capacitance information, and only one program [1] attempts to extract interconnection resistance information. To locate circuit problems, the IC designer must extract a complete spectrum of circuit parameters including interconnection resistance, ground capacitance (capacitance to the substrate), inter-nodal capacitance (or coupling capacitance), transistor sizes, and transistor areas. This is particularly true for designers of "analog sensitive" circuits such as random access memories, sense amplifiers, or bootstrap drivers.

General numerical techniques are known for solving each extraction problem in the spectrum [4]. The term "general" in this case is attached to methods that work for arbitrary shapes. Some of the general techniques—most notably for resistance and inter-nodal capacitance—are limited to very small problems because of their need for vast computing resources. In order to extract larger designs, automated circuit extractors have been developed [5] which use simpler techniques that do not solve the general problem. While the techniques are usually good for long, rectangular field regions, they sacrifice considerable accuracy around irregular regions.

9

process. diffusion has the largest capacitance because of its proximity to the substrate: metal has the least. Table 2-2 gives some sample capacitance constants for the NMOS process. While metal and polysilicon capacitances are linear, the ground capacitance for diffusion is not. Diffusion capacitance effects arise from the voltage-dependent, space charge layer formed by the back-biased p-n junction region under the diffused conductor. The junction capacitance as a function of junction voltage, V_{DB}, is

$$C_j(V_{DB}) = C_{j0} \frac{Area}{\left[1 - \frac{V_{DB}}{\varphi_B} \right]^w} .$$

The process dependent parameters—zero-bias junction capacitance, C_{j0}, bulk potential, φ_B, and junction grading coefficient, w—are extracted during model parameter extraction. Only the layout dependent $Area$ parameter comes from circuit extraction. The voltage-dependent capacitance value must be calculated in the circuit simulator, since V_{DB} is a function of the simulation. In the circuit simulators, a back-biased diode properly models the voltage-dependent diffusion capacitance described in the above equation. Generally, two parallel, back-biased diodes are needed: one for diffusion bottom capacitance, the other for diffusion edge (or *sidewall*) capacitance. We can see that capacitance modelling with back-biased diodes requires two circuit parameters, $Area$ and $Perimeter$, and five model parameters, C_{j0}, w, C_{j0sw}, w_{sw} (the sw subscript is for "sidewall" parameters), and φ_B.[3] Standard EXCL extraction models provide a switch that allows the user to enable either an extraction model with accurate diode capacitance modelling or an extraction model with approximate linear capacitance modelling.

2.4.2 Inter-Nodal Capacitance

We have seen in figure 2-6 that the *inter-nodal* or *coupling* capacitance lies between two IC conductors. The two conductors can be on any pair of conductors and can have many orientations between them. Particularly strong capacitive coupling exists between overlapping conductors (such as metal over polysilicon), or between long stretches of parallel conductors. However, the extractor should be prepared for any conductor orientation. Since silicon dioxide separates all conductors, inter-nodal capacitors are linear. The only exception to this is between diffused conductors, where inter-nodal capacitance is minimal.

---

[3] These model parameters are exactly the values included in the SPICE MOSFET model parameters, since SPICE includes diffusion capacitance if source or drain dimensions are specified.

2.4 Interconnection Capacitance Modelling

Two electrical parameters, resistance and capacitance, are extracted from layouts. To date, capacitance has been the more important for calculating circuit speed. Capacitive loading from interconnections frequently exceeds the capacitive loading from transistor gates.

2.4.1 Ground Capacitance

The total capacitance around a conductor is broken into the three components shown in figure 2-6. *Inter-nodal* capacitance forms between two conductors. The other two capacitances, *edge* and *bottom* capacitance, connect between the conductor and substrate. Since the substrate voltage remains fixed, these two capacitors are effectively "grounded"; the sum of edge and bottom capacitance is, therefore, known as *ground* capacitance. The dividing line between edge and bottom capacitor regions is only loosely defined by the plane extending straight down from the conductor edge. For a tighter definition, bottom capacitance is the portion of ground capacitance that is a function of conductor area (parallel to the substrate), and edge capacitance is the portion of ground capacitance that is a function of conductor perimeter. One can obtain good measurements for edge and ground capacitance with carefully selected test structures. One test structure has a very large circular conductor region. From this structure one measure mostly bottom capacitance. The other test structure has the same conductor surface area, but is arranged as a mesh of narrow conductor strips. The perimeter capacitance, which is no longer negligible, is calculated by subtracting the area related capacitance of the first test structure from the total measured capacitance of the second test structure.
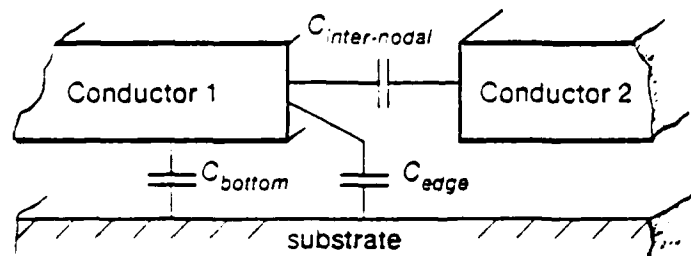
Figure 2-6: Capacitance types for interconnections.

Each conductor layer has a different capacitance per unit dimension. In our typical NMOS

	C_{gate} [2]	$C_{GS, e}$	$C_{GD, e}$
Actual	$2 (L^2 C_{ox})$	$1 (L x_d C_{ox})$	$3 (L x_d C_{ox})$
$L \times 1.5L$ rectangular	$1.5 (L^2 C_{ox})$	$1.5 (L x_d C_{ox})$	$1.5 (L x_d C_{ox})$
Solution (a)	$2 (L^2 C_{ox})$	$1.5 (L x_d C_{ox})$	$3 (L x_d C_{ox})$
Solution (b)	$2 (L^2 C_{ox})$	$1.5 (L x_d C_{ox})$	$3 (L x_d C_{ox})$

Table 2-1: Capacitance values for non-rectangular transistor

---

[2] The value of C_{gate} includes all capacitance associated with the channel. This includes the voltage dependent $C_{GD, i}$, $C_{GS, i}$, $C_{BD, i}$, and $C_{BS, i}$.

intrinsic part
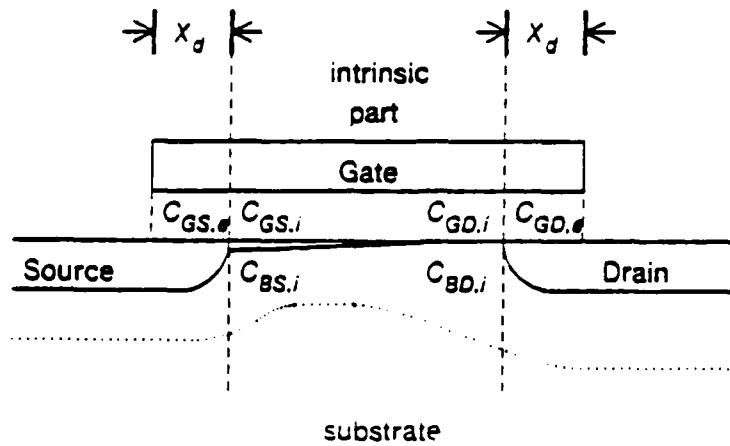
Figure 2-4: MOS capacitance model.

(a)

(b)
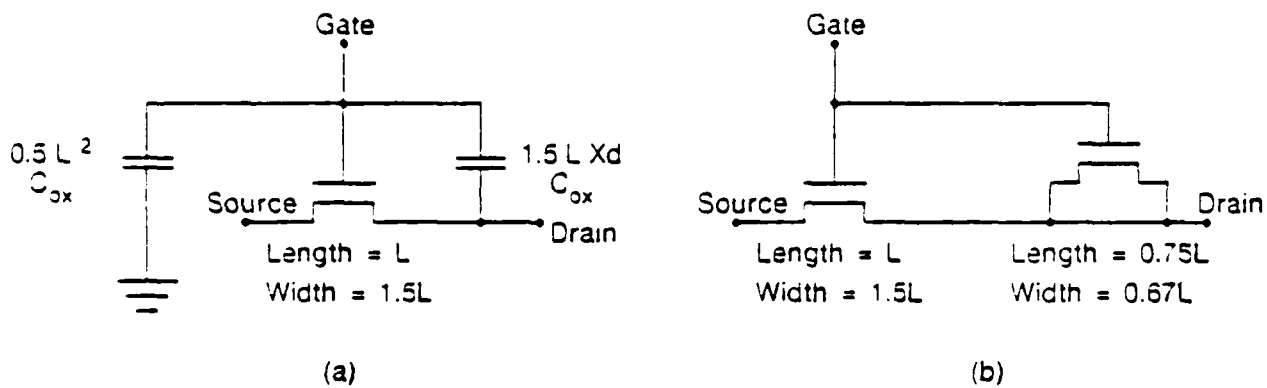
Figure 2-5: Modelling corrections for non-rectangular transistor.

(a) Linear capacitors for total gate capacitance and gate-to-drain capacitance guarantee a correct loading on the gate node. (b) To approximate channel capacitance effects, an MOS capacitor is added.

the MOS capacitance model shown in figure 2-4. Table 2-1 indicates how the capacitances are incorrectly estimated by the rectangular transistor. The total gate capacitance—a function of transistor area—is underestimated by 25%. The transistor capacitance problem is solved by adding other capacitive circuit elements. One of the solutions presented in figure 2-5 has two linear capacitors that correct for the total gate capacitance and the extrinsic gate-to-drain capacitance: the other solution has an "MOS capacitor" that corrects for the same capacitances, and includes a more accurate modelling of transistor channel charge. Either of these solutions can be included in standard extraction models of EXCL. The overestimated, extrinsic gate-to-source capacitance could be corrected with a negative capacitance. Since this element is unavailable in SPICE, it is best ignored.
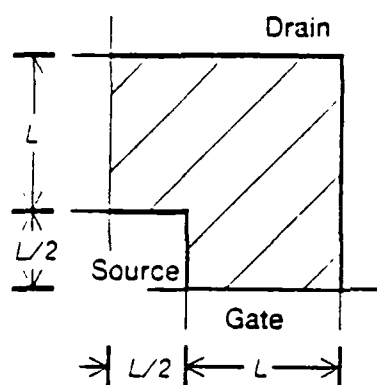
Figure 2-3: Top view example of non-rectangular transistor.

19

2.3 Transistor Modelling

The information found by a circuit extractor is most often used by a circuit simulator. The extractor's output must match the input expected by the simulator. Although transistors are the hardest element for circuit simulators to model, the transistor information required by circuit extractors such as SPICE [9] is not difficult to extract on a per transistor basis. The required circuit information for a bipolar transistor, for instance, amounts to no more than a transistor area. For a MOSFET transistor, the SPICE circuit simulator needs a length and width dimension. (All other parameters on a SPICE MOSFET "card" model the interconnections leading to the actual transistor terminals. EXCL does not use these, as it accounts for interconnection effects with lumped circuit elements as will be described soon.) To improve the transistor characterization, EXCL may extract additional circuit elements from the mask description of the transistor.

The SPICE circuit simulator also requires model information for each transistor type—model information like threshold voltage, transconductance, surface potential, etc. It is not the responsibility of a circuit extractor to find this information. Transistor *model parameter extraction* should be performed on a per fabrication technology basis, not on a per layout basis. Programs have been developed to assist in model parameter extraction from a set of MOS transistor curves [10].

We will now consider the MOS transistor. Aside from the MOS transistor's length and width, a circuit extractor must find which transistor model to assign to each transistor. The extractor must distinguish between depletion and enhancement transistors. Since short channel effects are, on occasion, poorly modelled by the circuit simulator, separate models may be needed for short-channel transistors. Therefore, the selection of transistor model depends not only on the occurrence of the ion implant mask, but also on transistor length. One should also note that the effects of length and width are non-linear especially at small dimensions, and that the extractor must find exact dimensions.

When a circuit simulator creates an internal equivalent model for a transistor, it assumes a rectangular transistor. If the actual transistor layout is not rectangular, slight simulation model errors will be present. In our extraction model, the transistor's length and width are selected to give correct current conduction modelling. For the non-rectangular transistor of figure 2-3, for instance, we choose a length, L, and an approximate width, $1.5L$, for correct conduction modelling. However, the transistor capacitances are not accurately modelled. This discussion of transistor capacitance uses

18

Noise Immunity   The switching transfer characteristics found with the "logic level" parameters are important in finding a logic element's noise immunity. To find sources of coupling noise requires coupling capacitances and ground capacitances.

Power Consumption

Supply current calculations require the same static circuit elements which are needed for logic level calculations.

Charge Leakage   Charge leakage usually occurs at a rate slow enough to make capacitances unnecessary. A main source of leakage is through the back-biased diode of diffused conductor regions.

A designer should make an even more basic check to verify correct logical operation. A switch-level representation is the most convenient form of portraying the logic elements of an MOS circuit. The switch-level representation views all transistors as a switch between the source and drain, denoted here as "node 1" and "node 2".[1]   The switch is either "off" (non-conducting), "on" (conducting), or in the "x" (unknown) state, depending on transistor type, and the logic level of the transistor's gate. In most switch level representations [7, 8], the switch's "on" conductance between "node 1" and "node 2" is also relevant. This is especially true for ratioed circuit designs in which two or more "on" transistors pull a single node in opposite directions. Finally, the MOS switch-level model must include information about node ground capacitances in the event of charge sharing. Charge sharing occurs when two otherwise isolated nodes become connected though a transistor. If the nodes start at opposing logic levels, then the resulting logic values on both nodes take on the initial value of the "strongest" node, where "strongest" means greater capacitance to ground. Two charge sharing nodes with roughly equivalent strengths acquire "x" states on both. An extraction model for switch-level simulation locates transistors with conduction and capacitance information, and locates interconnection ground capacitances. Since the switch-level simulation has only a few discrete states, careful calculation of these parameters is not needed.

---

[1] Because the current flow direction changes for some transistors of an MOS circuit, and since the model views the device reciprocally, the standard node designations, "source" and "drain", will not be used.

17
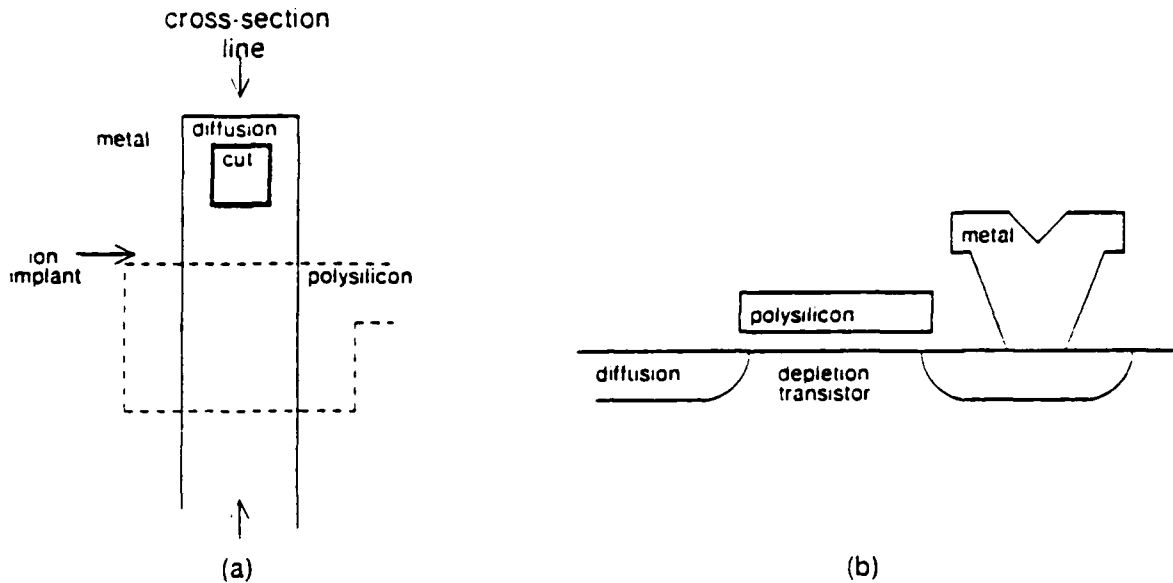
Figure 2-2: NMOS technology

(a) sample layout, (b) IC structure cross-section of sample.

2.2 Types of Circuit Analysis

A circuit design passes through many representation levels: from the architectural, register, logic, and transistor levels to, finally, the mask layout level. After the designer progresses through all these levels and has a mask layout, the circuit extractor enables him to check whether the hardware described by the mask layout will perform as expected. Performance checks are usually made to ensure adequate circuit speed, logic levels, noise immunity, and power distribution. For dynamic MOS circuits, charge leakage might be checked. Each of these checks needs a different set of circuit parameters. NMOS circuit parameters are matched with analysis types in the following paragraphs.

Speed — Delays in an MOS circuit is dominated by RC time delays associated with the transistors and interconnections. All transistor conduction and capacitance effects are important, as are interconnection resistance and ground capacitance effects. Inter-nodal capacitance is important only to the extent that it adds to total capacitance.

Logic Levels — Guaranteeing adequate logic levels requires only DC or static checks. Therefore, capacitances are not needed—only transistor conduction parameters, and power supply line resistances.

16

thick silicon dioxide layer over most of the IC surface. but over regions specified by the **diffusion** mask the polysilicon rest on a much thinner silicon dioxide layer. Afterward. when diffusion conductors are created. the polysilicon blocks the diffusion impurities from entering the substrate at the active areas. This combination of thin oxide and no diffusion impurities yields MOSFET transistors. The transistor's source and drain are located at either edge of the active area where **diffusion** extends beyond the overlap: the gate is the polysilicon conductor.

The process engineers selects the proper semiconducting material types for the silicon substrate and diffusion impurity such that the MOS transistors are N-channel (that is the majority carrier in the channel is electrons). By selectively regulating certain minute amounts of impurity in the active area. the technology provides two types of N-channel transistors. One type. the *enhancement* transistor. has a positive threshold voltage and can therefore be turned off completely. These transistors serve as switching elements. The other *depletion* transistor has a negative threshold. and thus always conducts to some degree. These are used as "pullup" devices for the NMOS logic gates. The layout designer discriminates between the two transistor types with an additional **ion implant** mask that is placed over depletion transistor areas.

The IC layout designer must follow a set of *design rules* governing the minimum dimensions for layout patterns. The rules specify such parameters as minimum conductor sizes. spacings. extensions. etc. The design rules are based on the fabrication and lithography process and are set such that one can be relatively certain of obtaining a circuit free of electrical defects. For the most part. the extractor is unaffected by the occurrence of a design rule violation in a layout. Unlike the photolithographic process. the extractor can separate lines to a much higher degree. Nonethless, such designs should not be encouraged. Some very contorted layout patterns that violate one or more design rules cause the extraction model to fail and generate wild circuits. For this reason, the extractor expects an input layout that violates no design rules.

Figure 2-2(a) shows a sample NMOS layout. It is accompanied by the corresponding cross-sectional view of the IC structure. For all layouts illustrated as examples in this document. layers are identified by name when relevant.

and insulating material. Active device modelling is less general. and depends more on the fabrication technology. For the discussion of active device modelling and for all of the examples. we will assume a fixed NMOS technology. The mask sequence for our simple NMOS technology is the same as that described in Mead and Conway [6]. The next section briefly outlines the technology.

2.1 Nmos Technology

The simple NMOS technology has three conductor types. diffusion. polysilicon. and metal; each occupies a different vertical position or "layer" (see figure 2-1). Two of the layers, diffusion and polysilicon. are positioned above the silicon "substrate". and are surrounded by insulating silicon dioxide. sometimes called "oxide". The third conductor. diffusion. exists as part of the substrate itself. isolated electrically by an impurity-induced. back-biased diode. The IC designer patterns the three conductors in the two dimensions parallel to the substrate. The complete set of patterns for all layers is the *layout*. No electrical conduction exists between the three conductors. and the designer is free to cross connectors on different layers as he pleases. with the exception of diffusion and polysilicon. The designer can. however. provide an electrical connection between metal and either of the other layers by placing a "contact cut" at any horizontal position occupied by both layers.
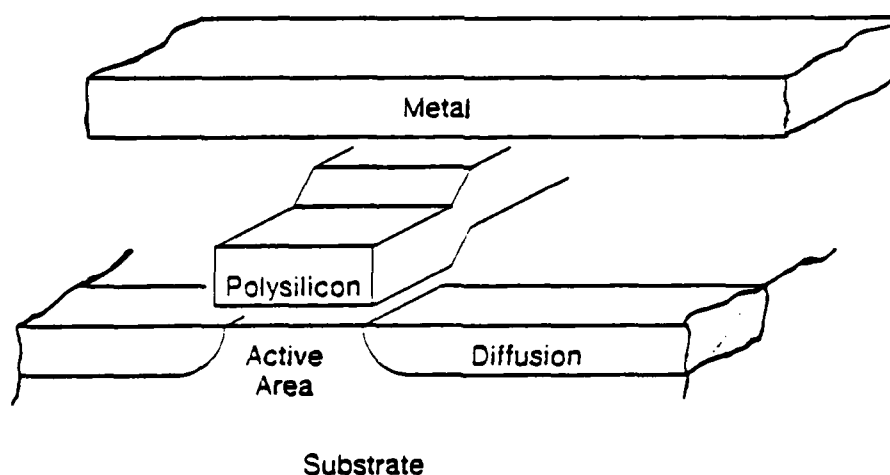
Figure 2-1: Conductors in the NMOS technology

Positions on the layout with both diffusion and polysilicon mark the active regions for this technology. During IC fabrication. the polysilicon layer is deposited first. The polysilicon rests on a

14

CHAPTER TWO

Extraction Models for Integrated Circuits

An *extraction model* is the complete set of rules, algorithms, and constants that is applied to an IC layout to convert the mask information into an equivalent circuit. For any layout, an extraction model should generate a unique equivalent circuit, however, we have few restrictions on which rules and algorithms we can actually place in the extraction model. An extraction model generally reflects the intended use for its output. For instance, if we intend to simulate the output circuit with a logic simulator, the extraction model should contain rules for finding logic gates and interconnections between logic gates. With EXCL, our primary wish is to extract enough circuit information such that we can accurately compute the anticipated performance of the integrated circuit on a continuous voltage, current, and time scale. For digital IC's, this means that we do not intend to deal with simple, discrete logic levels, but with the complete range of analog voltages.

One circuit performance behavior usually sought by the digital IC designer is the circuit's switching speed, but, he may also wish to characterize coupling noise, logic levels, power consumption, leakage current, ...; the list goes on. When characterizing each of these circuit behaviors, different circuit elements become important. While one exhaustive extraction model suffices for all of these circuit analyses, we can take a more efficient approach and change the extraction model to fit the type of circuit analysis. EXCL has the capacity to incorporate broad changes in the extraction model. In this chapter we will examine which circuit elements are relevant for each type of analysis, and will examine the different extraction models for an example technology. Integrated circuit modelling can be categorized into two main areas—the modelling of active devices, and the modelling of interconnections, and each will be described in this chapter.

All of the discussion in this document assumes a planar IC technology, but beyond that technology restriction, the principles can be applied to most any technology. This is particularly true for interconnection principles, because interconnection models are similar for any type of conducting

13

developed for EXCL. The basic algorithm for dividing a region into its subregions with different solution techniques is similar for each of these problems. The main discussion of the algorithm is found in the resistance chapter (chapter 4).

geometric file

geometric modules		resistance extractor

EXTRACT ← EXTRACT·LIBRARY ← coupling capacitance extractor

| circuit formatting modules | | transistor size extractor |

circuit file

Connectivity Extractor   Extraction Library
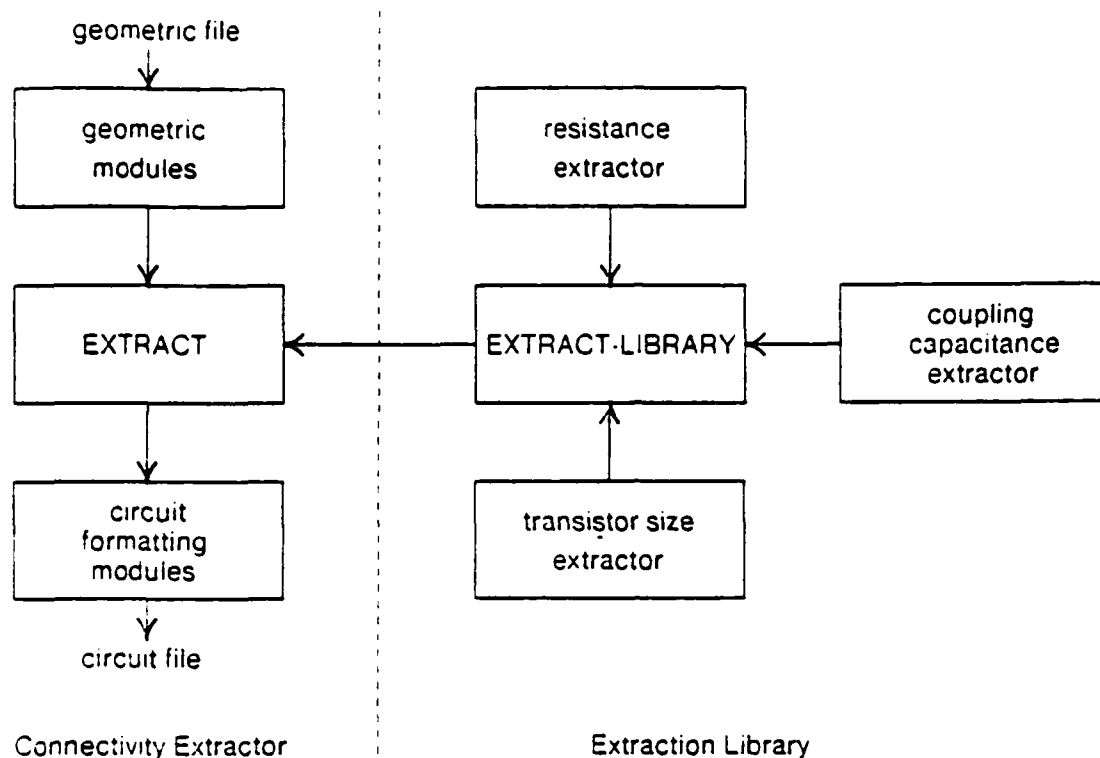
Figure 1·1: Organization of EXCL

1.2 Overview of Remaining Chapters

The thesis is presented in three main parts. The first part (chapter 2) introduces an NMOS fabrication technology, and discusses each of the relevant circuit parameters that might be extracted. This part presents some of the basic issues that a designer must consider when developing an extraction model for EXTRACT.

The next part (chapter 3) discusses the connectivity extractor of EXCL. It discusses the geometric and circuit portions of the connectivity extractor, including the two technology dependent modules.

The last main part (chapters 4 through 6) presents in detail each of the extraction algorithms for extracting resistance, interconnection capacitance, and transistor sizes. Some of the extraction algorithms are well-known, while others—most notably for coupling capacitance—had to be

11

This thesis describes an automated circuit extractor, EXCL (EXtractor of Circuits from Layouts), that extracts complete interconnection and transistor information. It uses a range of extraction techniques for each problem—some are special-case and fast, others are general and slow. In the resistance, inter-nodal capacitance and transistor sizing problems, a single and powerful algorithm separates field regions into *subregions* of three kinds. Where the fields are one-dimensional (as the fields describing conduction in a long straight wire) one kind of subregion is formed, and the problem is solved using a simple equation. Of the remaining subregions, those with prespecified, commonly-occurring shapes have their solution found in a library. Only those subregions that cannot be solved with the previous two techniques use general techniques. Separating the problem in this way allows EXCL to operate with reasonable speed without sacrificing accuracy. This gives EXCL the capacity to detect potential circuit hazards in larger IC designs containing sensitive circuits.

1.1 Overview of EXCL

EXCL converts raw mask geometric data generated by the IC designer into an equivalent circuit representation for subsequent simulation or analysis. An important feature of EXCL is its programming modularity. The program is not bound to any specific IC fabrication process or mask set specification, for all technology related instructions reside in two easily-modified program modules.

Figure 1-1 shows the general organization of EXCL. It is broken into a *connectivity extractor* part and an *extraction library* part. The connectivity extractor processes geometric mask information into isolated regions associated with individual circuit elements. This is controlled by one of the technology dependent modules. Next, the other technology dependent module, EXTRACT, instructs EXCL on how to convert geometric data into circuit data. It does so by calling on extraction algorithms contained in the extraction library. For instance, the user can include a command in EXTRACT that says: for each region of a certain layer, activate the extraction library's "resistance extractor" to convert geometric data into resistor network data. By allowing the user to make such decisions, he can tailor the extractor to meet his own needs, i.e., he can make his own *extraction model.*

Each of the basic extraction problems is self-contained in the extraction library. The library includes procedures for computing resistance, coupling capacitance, transistor sizes, and other parameters based on a region's area or perimeter.

10

Capacitance	Layer pair	
C_{bottom}	diffusion, substrate	1.25×10^{-4} pF / μm^2
C_{edge}	diffusion, substrate	3.5×10^{-4} pF / μm
C_{bottom}	polysilicon, substrate	0.50×10^{-4} pF / μm^2
C_{edge}	polysilicon, substrate	0.40×10^{-4} pF / μm
C_{bottom}	metal, substrate	0.25×10^{-4} pF / μm^2
C_{edge}	metal, substrate	0.40×10^{-4} pF / μm
$C_{inter\text{-}nodal}$	diffusion, metal	0.25×10^{-4} pF / μm^2
$C_{inter\text{-}nodal}$	polysilicon, metal	0.30×10^{-4} pF / μm^2

Table 2-2: Sample capacitance constants for NMOS process
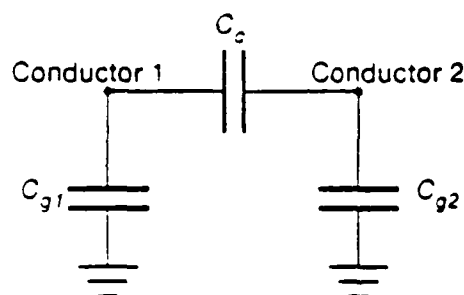
Figure 2-7: Coupling capacitance between two nodes.

Inter-nodal capacitance values are used for circuit noise and speed analyses. When two nodes are coupled with an inter-nodal capacitance, C_c, as shown in figure 2-7, a change on one node induces a voltage on the coupled node. In IC's, the induced voltage is coupling "noise". If the conductor 1 voltage changes by ΔV_1, the induced voltage change on conductor two is

$$\Delta V_2 = \Delta V_1 \frac{C_c}{C_c + C_{g2}} . \tag{2.1}$$

For digital IC designs, we can determine the maximum induced noise on conductor 2 by assuming a maximum voltage change on conductor 1. The maximum ΔV_1 equals the difference between the high

24

and low logic voltages, $V_{high} - V_{low}$. If we are only interested in noise values greater than a minimum, $\Delta V_{2,min}$, then using equation (2.1), we know that the following condition must hold:

$$C_c > \frac{\Delta V_{2,min}}{V_{high} - V_{low}} (C_c + C_g) = \gamma (C_c + C_g). \quad (2.2)$$

We can effectively compute γ by knowing the noise immunity characteristics of the logic circuits. A large number of the coupling capacitances in an LSI circuit fail the condition of equation (2.2), for potentially every node pair has some inter-nodal coupling. An extraction model that recognizes the condition, will be vastly more efficient. Inter-nodal capacitance effects speed analysis only by contributing its capacitance to a node's total capacitance. Activating the condition of equation (2.2) interjects a maximum error of γ into the speed calculations.

2.5 Interconnection Resistance

Due to the uniform thickness and resistivity of integrated circuit conductors, the resistance problem is usually a two-dimensional, geometric one. In the well-known resistance equation,

$$R = \rho \frac{Length}{Width \cdot Thickness} ,$$

the known conductor thickness and material resistivity combine to give a new parameter, *sheet resistance* or $\rho_{sh} = \rho/Thickness$. Resistance is now described by

$$R = \rho_{sh} \frac{Length}{Width} .$$

The dimension of ρ_{sh} is ohms, but it is typical to refer to the dimension as ohms per square ($\Omega/\square$), since the ratio *Length/Width* gives the number of end-to-end "squares" for a conductor. Table 2-3 lists some sample sheet resistances for the NMOS process.

Conducting Layer	Sheet Resistance
diffusion	12 $\Omega/\square$
polysilicon	25 $\Omega/\square$
metal	0.03 $\Omega/\square$

Table 2-3: Sample sheet resistances for NMOS process

The resistance of an interconnection is calculated between different *connections* (contact

cuts, transistor terminals. etc.). Chapter 3 gives a complete list of connection types. Some connections are single points, but the more interesting types extend over a large area; most notable in this category is the transistor channel to diffusion conductor connection. Usually, the resistivity on one side of the connection (channel) is much larger than the resistivity on the other (diffusion). In such cases, the <u>tangential</u> resistance is unimportant on the high-impedance side and important on the low-impedance side. Assuming a uniform voltage on the connection, the tangential resistance is modelled adequately when the full connection region is present on the high-impedance side. On the low-impedance side, however, separate connections are needed to account for tangential resistance (see figure 2-8).
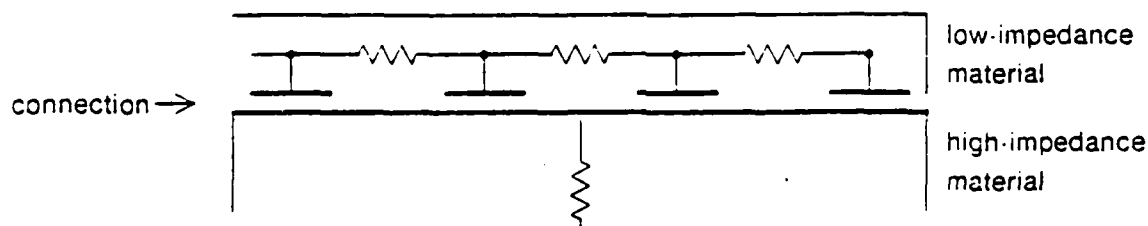
Figure 2-8: Connections for modelling tangential resistance.

2.6 Modelling Distributed RC's

Both the resistance and capacitance effects distribute over the length of an IC interconnection. While some simulators can estimate delays directly for distributed RC lines [11], most simulators cannot. For such simulators, an extractor must generate an equivalent resistor and capacitor network with discrete nodes and elements. In EXCL, the resistance and capacitance extractors combine to model distributed RC's with an n-stage π-ladder network, as shown in figure 2-9. The total line resistance and capacitance are denoted by R and C, respectively. In addition to R and C, IC interconnections usually have a discrete drive resistance, R_T, and a discrete load capacitance, C_T, connected at opposite ends of the interconnection. R_T and C_T are shown for the one-stage π-ladder. As the number of π-ladder stages increases, the modelling becomes more accurate. EXCL always breaks a resistive line and inserts a node at an interconnection branch, but to guarantee a certain modelling accuracy, long interconnections without branches may need added nodes to increase the number of ladder stages. For each node pair of the π-ladder network, EXCL inserts the whole, extracted resistance between the nodes and divides equally the extracted capacitance from the physical region between the nodes.
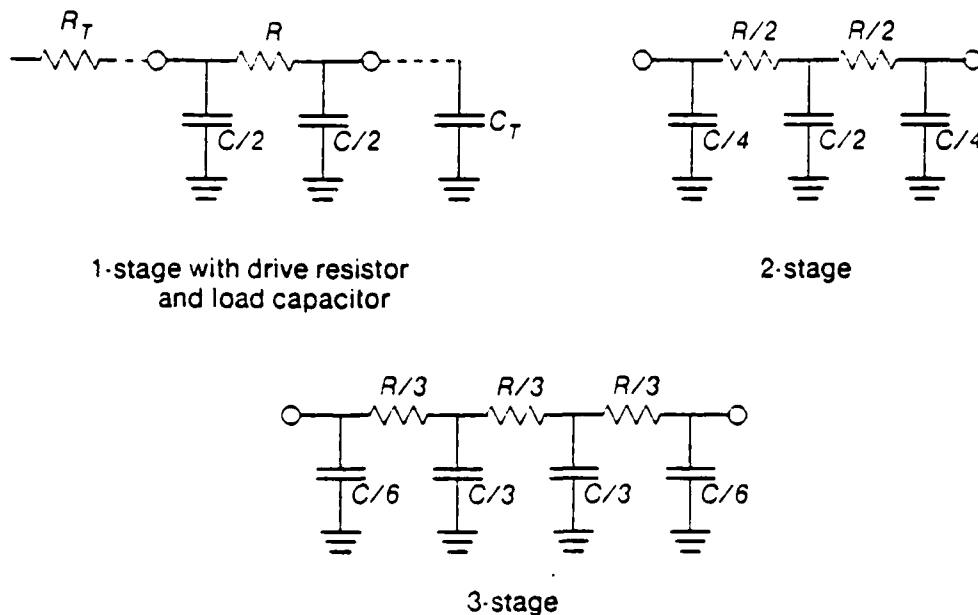
Figure 2-9: Pi-ladder networks for approximating distributed RC lines

The ladder network does not model the interconnection exactly, and how close the network approximates the actual behavior is discussed by Sakurai [12]. Appendix A calls upon these results and develops a criterion for the number of π-ladder stages needed. The criterion guarantees that the ladder network step response time does not vary by more than $\Delta t_{0.9}$ from the true distributed RC step response time. The time considered here measures the delay for voltage at the end opposite the step source to reach 90% of its final value.

2.7 Modelling Fabrication Degradations

The true physical regions on an IC differ in detail from the regions of a mask description. Errors with photolithography may cause real objects to expand or shrink from the mask specifications, or errors with mask alignment may cause an overall offset between two or more mask layers. These errors may alter interconnection resistance, inter-nodal capacitance, and even transistor size by a noticeable amount. To model the true behavior of the IC, the extraction model should simulate these anomalies by translating or expanding (shrinking) all rectangles of a layer by equal amounts.

27

CHAPTER THREE

Connectivity Extraction Algorithms

The connectivity extractor is divided into four subprograms. CIFPARSE and SORTREC execute first during an extraction and preprocess the geometric IC mask data. CONNECT, the third and main subprogram, converts geometric mask information into an internal circuit representation. Lastly, the fourth subprogram, FORMAT, converts the internal circuit representation into the appropriate output network format. The algorithms of each subprogram that pertain to extracting connectivity information from a layout are described in detail in this chapter.

3.1 Decomposition of Mask Geometries

The first subprogram, CIFPARSE, parses the geometric mask description provided by the user. The most common and universal source is a Caltech Intermediate Form (CIF) [6] file. A CIF file is a collection of *symbols* describing IC geometric layouts. A symbol may contain *mask objects* such as rectangles, wires, polygons, point names, etc., each tagged with its mask layer. A symbol may also contain *symbol calls* to other, previously defined symbols.

CIFPARSE fully instantiates the geometric mask description into each of its component *boxes* and *named points*. Interiors of Boxes define the areas of interest for a given mask layer. A box is described by four integers representing the minimum and maximum *x* and *y* coordinates, and by one character representing the mask layer. A named point given by the user tags a mask region with a meaningful name; it is primarily used to tag a name to an electrical node. A named point is described by a character string for the name, and by an *x* and *y* coordinate and mask layer which locate the box region. After instantiation, these are the only geometric object types used by EXCL.

During the CIF instantiation process, all wires and polygons must be converted to an equivalent set of orthogonal rectangles, and all symbol calls must be replaced by the actual mask objects contained in the called symbol. Only named points from the top-level CIF symbol are retained to avoid

28

name conflicts arising from multiple calls to lower-level CIF symbols. Thus, after expansion the resulting data structure contains a "smashed" set of all boxes describing the layout and a set of named points from the top-level CIF symbol.

By instantiating the entire layout, we note that the layout hierarchy is lost, resulting in wasteful, repeated extractions of cell layouts which are replicated. A "hierarchical extractor"—that is an extractor that recognizes cell replications, and extracts the layout only once—would not only save extraction time, but could also pass the layout hierarchy through to the circuit network. The problem with this approach is that typically no restrictions are placed on the cell's layout boundary, thus allowing arbitrary overlaps of cells. (Allowing no overlaps, on the other hand, is too restrictive.) An erroneous overlap of cells could alter the intended network into something quite different. This IC design disaster must be detected by the extractor. Clearly, when a general hierarchical extractor looks for repeated cells, it must also examine the overlaps of other cells for each replication. This adds considerably to the extractor's execution time and complexity. A better approach is to support the layout and circuit hierarchy in a system at a higher level. Regulating and checking cell overlaps would be carried out in the higher-level system.

A second problem with converting the mask data to boxes is that of non-orthogonal line definitions. Handling non-orthogonal geometries properly necessitates additional complexity in the geometric data structures and procedures—complexity that has questionable value when considering the great infrequency of non-orthogonal geometries in IC layouts. In an alternate approach, a non-orthogonal geometry is converted into a set of smaller, orthogonal rectangles that approximate the original geometry as figure 3-1 demonstrates. Geometric computations are affected as follows:

1. Calculations of area are no different than the actual values.
2. Calculated perimeter values are larger than the real values due to the staircase effect of the approximating rectangles.
3. Resistance calculations may be different. Non-orthogonal geometries always use general calculation methods. We will see in the next chapter that such methods divide the geometry into a square grid. If the staircase dimensions are small enough to match the grid spacing, resistivity calculations are not effected.

original
edge

rectangle
approximation
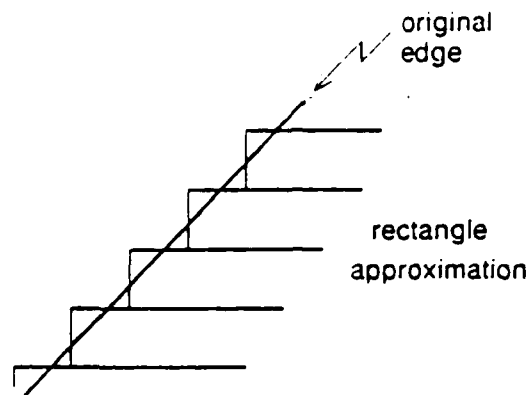
Figure 3-1: Approximation for shapes with non-orthogonal edges

3.2 Sort by Maximum Y

In preparation for the scanning process of CONNECT, the second subprogram, SORTREC, sorts the geometrical objects from maximum y coordinate to minimum y coordinate. A box's y coordinate is considered as the top of the rectangle. If one covered a plot of the layout with a sheet of paper and slowly pulled it down, the order of appearance of objects has the same ordering that results from SORTREC.

3.3 Scan of Geometrical Objects

Once the entire layout has been decomposed into sorted mask objects, the next process is to locate the regions of relevant circuit elements defined by rectangle overlaps and to group together rectangles that are connected. This is the job of the third and major subprogram of the connectivity extractor. Figure 3-2 charts the major modules of the third subprogram. Each box represents (1) a collection of one or more procedures, and (2) in many cases, a data type upon which the procedures operate. The lines show the directions of information flow.

Basically, this subprogram, *examines* the mask objects sequentially from the geometric input file, $\mathcal{G}$. When an object is examined, relevant intersections with other mask objects in a "scan-view set", $\mathcal{V}$, are located and remembered. Then, the newly examined object is added to set $\mathcal{V}$ and the cycle is repeated with the next mask object from the input file. This can be a costly portion of connectivity

30
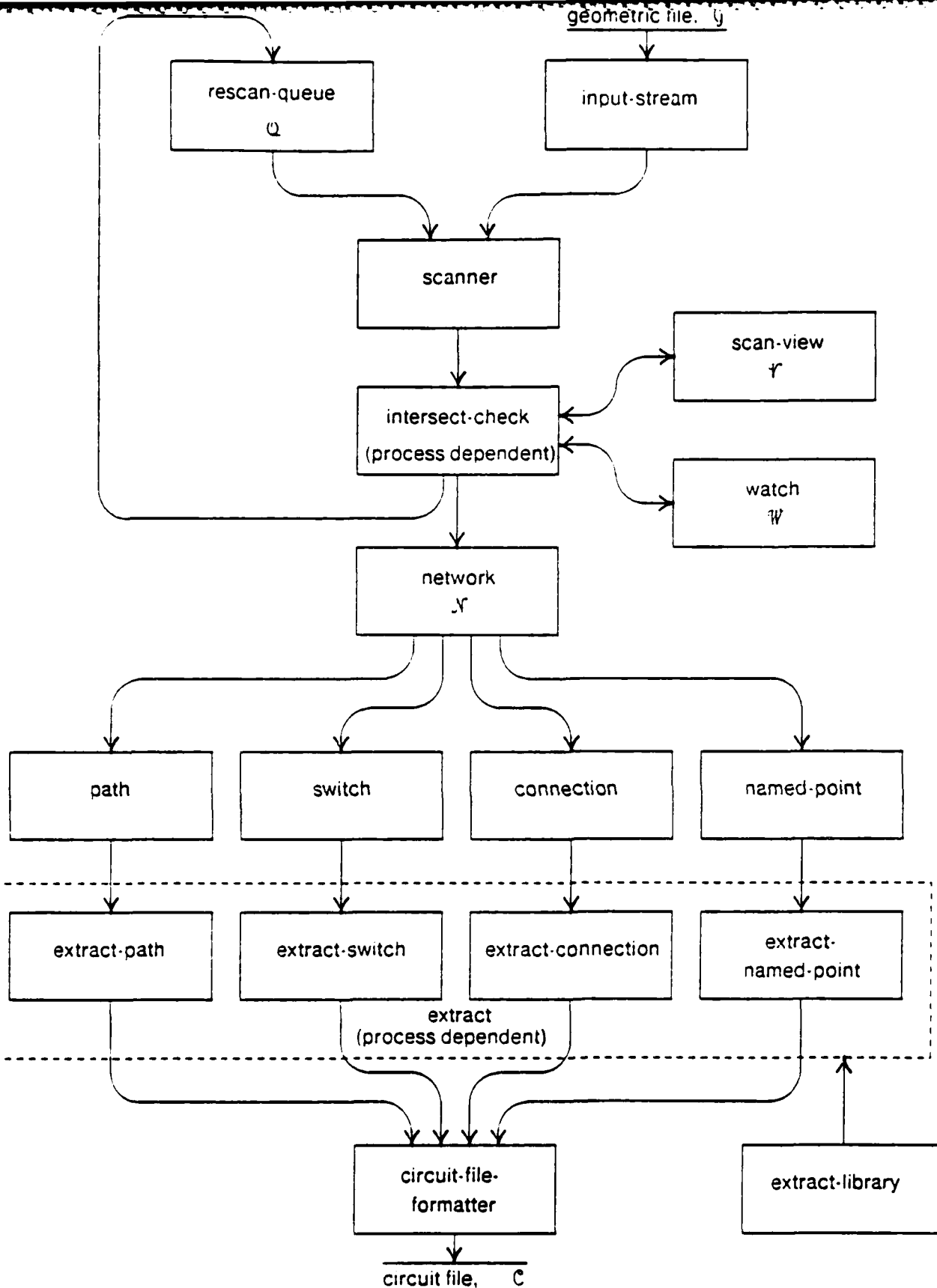
Figure 3-2: Major Program Modules of CONNECT

extraction if the mask object examination is done in a random order. The set Γ continues to grow with each new object, and the complexity increases as the square of the number of rectangles.

3.3.1 Reducing the Vertical Search

The scanning process used by EXCL and other extractors [2] reduces execution time. A horizontal scan-line begins at the top of the layout. As the sorted mask objects are examined, the scan-line is always defined as the top edge of the mask object. Clearly, from the nature of the sort described in the previous section, the scan-line always moves downward. With the sorted scan, mask object intersection checks, need to be done only with rectangles still lying in the range of the scan-line. Thus, when the scan-line moves below a mask object in the "scan-view set", Γ, it is removed from further intersection checks. The execution time for this algorithm depends upon the number of objects at any given scan-line, and thus, roughly upon the width of the layout. For a square layout, the scanning process has a computation complexity approximately of order $N \cdot \text{Log}_2 N$ for N mask objects.

3.3.2 Reducing the Horizontal Search

The scanning process narrows the search for intersecting rectangles to the approximate vertical coordinate. Narrowing the search along the horizontal coordinate results in a further reduction of computation complexity [13, 14]. One can do this by placing objects from the set Γ into horizontal bins. A bin contains all mask objects of Γ which lie between two fixed horizontal coordinates, x_n and x_{n+1}. Thus, each new mask object needs to be checked against only those objects in bins which fall in the same x-coordinate range as the new object.

The procedure is complicated slightly since the objects may span more than one bin as shown in figure 3-3. The objects within a bin are further subdivided into two columns:

1. The objects whose left edges are in the range of that bin are placed in the *left edge* column,
2. all other objects are placed in the *non-left edge* column.

When the "new object" is checked for intersections, we wish to check all mask objects from the bins which lie in its x-coordinate range, but only once for each object. The objects in both columns are checked from the bin at the left edge of the new object (bin N in figure 3-3), while only the objects in the left-edge column are checked in the remaining bins (bins $N-1$ and $N-2$). For the example in

32
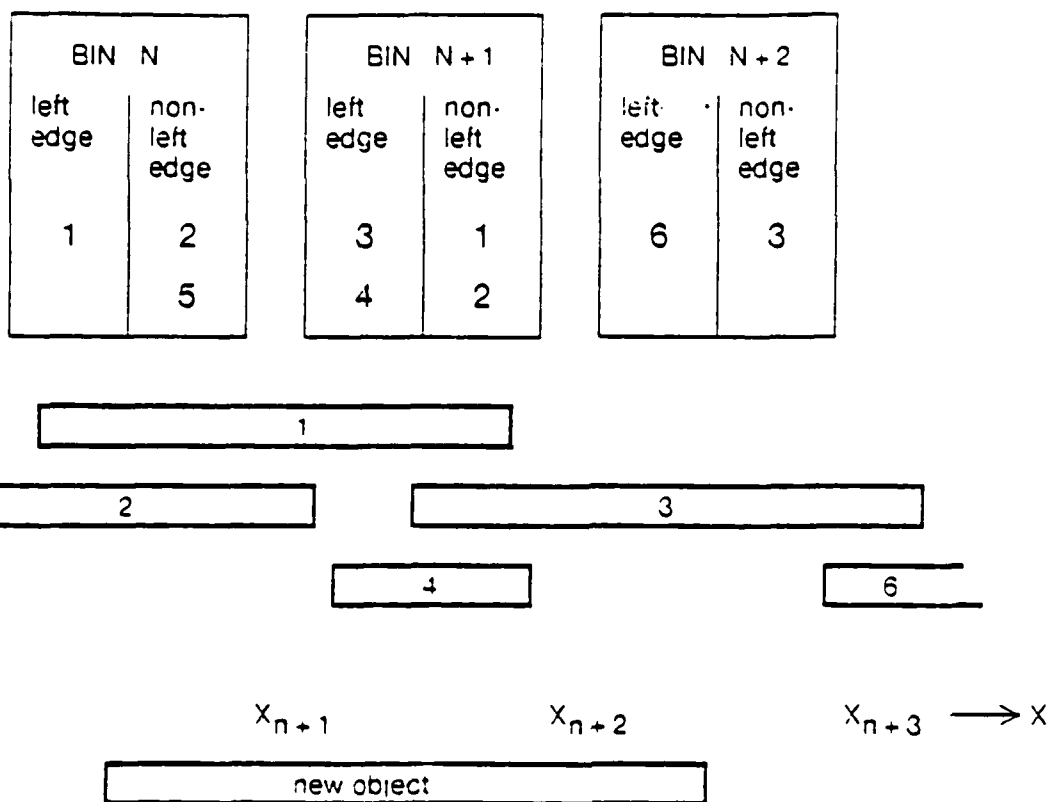
Figure 3-3: Bin Placement of Mask Objects

Only the x-coordinates of the mask objects are shown: all objects intersect with the scan-line defined by the y-coordinate of the new object.

figure 3-3, mask objects 1 and 2 are checked only at bin N, not bin $N+1$. Note also that the objects checked from bin $N+1$ (mask objects 3 and 4) need no further x-coordinate intersection check, but that objects from the bins at the left or right edge of the new mask object (all mask objects except 3 and 4) must be checked further. for some of them may not actually intersect (as in the case of mask objects 5 and 6).

3.3.3 Rescan Queue

Although coordinate information below the scan-line is known during the examination of a rectangle, care must be taken to assume nothing about the geometries or connectivity below the scan-line. In some cases, new rectangles are defined which lie below the scan-line and thus should be reinserted into the sorted input stream to be examined at a later time. Such rectangles are inserted into the *rescan-queue*, Q. Rectangles in the rescan-queue are treated as though they are merged back into the geometric input stream, G. To demonstrate how the rescan-queue might be used, consider the layout shown in figure 3-4.
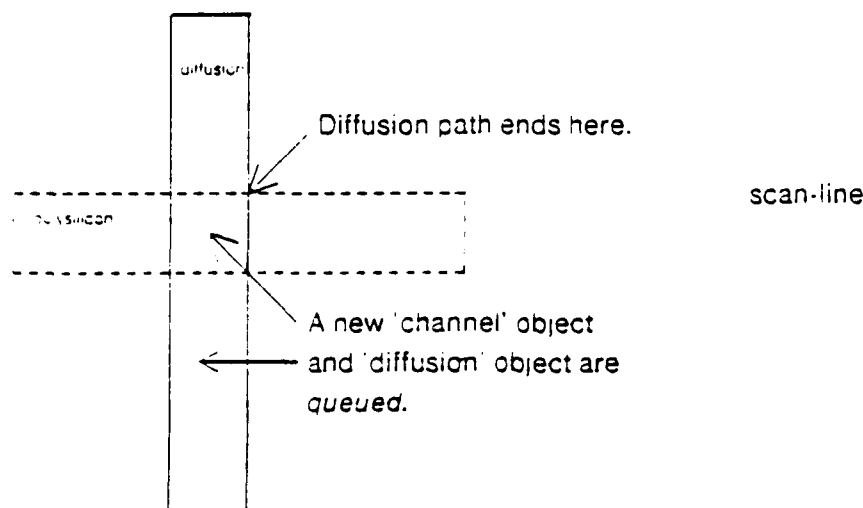
Figure 3-4: Scanning process of a MOS transistor layout

When the diffusion rectangle is first encountered, it is treated as a single rectangle. It is not until later, when the polysilicon rectangle is examined, that the extractor knows differently. The original diffusion rectangle becomes two diffusion rectangles (each a different node) and one "channel" rectangle. The extractor removes the original rectangle and replaces it by three rectangles: (1) the upper diffusion rectangle is added to the node of the original rectangle as though it has already been examined, (2) the channel rectangle and (3) the lower diffusion rectangle are added to Q, since they are on or below the scan-line. Details of the operation on diffusion and polysilicon rectangles will be discussed the next section.

Procedure SCANNER demonstrates the main points presented in this section. INTERSECT-CHECK and NETWORK-CLEAN will be discussed later. (In the notation used here, a box, $\Box$, precedes mask-object variables, and underlined identifiers are constants.)

34

```
procedure SCANNER (G):
begin
  N — empty-network;
  Q — empty-queue;
  Y — empty-set;
  W — empty-watch;
  next-clean — layout-top – clean-interval;
  for each □r ∈ G do
      rec-scan-line — □r.scan-line;
      while QUEUE.PEEK(Q).scan-line > rec-scan-line do
              □s — QUEUE.NEXT(Q);
              INTERSECT-CHECK(□s, Y, Q, W, N);
              end
      INTERSECT-CHECK(□r, Y, Q, W, N);
      if rec-scan-line < next-clean then
        NETWORK-CLEAN(N, rec-scan-line, C);
        next-clean — rec-scan-line – clean-interval;
        end
      end
  for each □s ∈ Q do
      INTERSECT-CHECK(□s, Y, Q, W, N)
      end
  NETWORK-CLEAN(N, layout-bottom, C):
  return(C)
  end
```

3.4 Geometrical Object Intersection

Up to this point, the program has made no distinction between the different layers of the mask objects. All operations have been independent of extraction-model or IC fabrication technology. This section discusses the procedure INTERSECT-CHECK, a single procedure where all process-dependent, geometric operations are defined. For instance, this procedure contains the rules governing which overlapping layers define a transistor, which rectangle combinations define a contact, and which rectangle groups define an interconnecting wire. It is one of the two extraction model dependent procedures.

INTERSECT-CHECK is called for each mask object that is examined during the scanning process. With each examined mask object, r, it updates the data structures by:

1. locating relevant intersections between r and other mask objects included in the "scan-view set", Y.

35

2. adding new mask objects appearing on or below the scan-line to the rescan-queue, Q
   (sect. 3.3.3),

3. adding new network information to N, and

4. placing r in the "scan-view set", V.

intersecting layers	consequential action	unilateral/ bilateral
Diffusion, Polysilicon	Channel — intersection requeue remove diffusion under channel	Bilateral
Diffusion, Diffusion	form interconnecting paths	—
Polysilicon, Polysilicon	form interconnecting paths	—
Channel, Channel	Combine channel regions	—
Channel, Diffusion	add switch node1/node2 connection	Bilateral
Channel, Polysilicon	add switch gate connection	Unilateral
Channel, Implant	set switch type to "Depletion"	Unilateral
Metal, Metal	form interconnecting paths	—
Cut, Metal	look for other side of contact	Unilateral
Cut, Polysilicon	form connection___ (requeue cut ∩ poly , if any)	Unilateral
Cut, Diffusion	form connection_____ (requeue cut ∩ diffusion , if any)	Unilateral
Named-Point, Diffusion	name path region	Unilateral
Named-Point, Polysilicon	name path region	Unilateral
Named-Point, Metal	name path region	Unilateral

Table 3-1: Intersection checks for example NMOS process

To introduce INTERSECT-CHECK, the example NMOS process is used. Table 3-1 lists all of the relevant intersections for the NMOS process. The second column lists the actions taken by INTERSECT-CHECK when an intersection is located, while the third column states whether the

intersection needs to be checked in one direction or both. A *bilateral intersection check* is one that must be made when either the first layer or the second layer is examined. Figure 3-5(a) shows two configurations for the bilateral intersection between Diffusion and Polysilicon. The intersection occurs in one case, while examining Diffusion, and in the other case, while examining Polysilicon. The Diffusion-Polysilicon intersection demonstrates a typical bilateral intersection— the two layers cross, each extending outward from the intersection. A *unilateral intersection check* is one that needs to be made only when examining a rectangle of one of the layers—in the example, the first layer listed in the table. It is typical of unilateral intersections for the region of one layer to completely enclose the region of the other. Figure 3-5(b) shows a unilateral intersection between Cut and Metal. For the most part, the type of intersection is determined by the process' design rules.
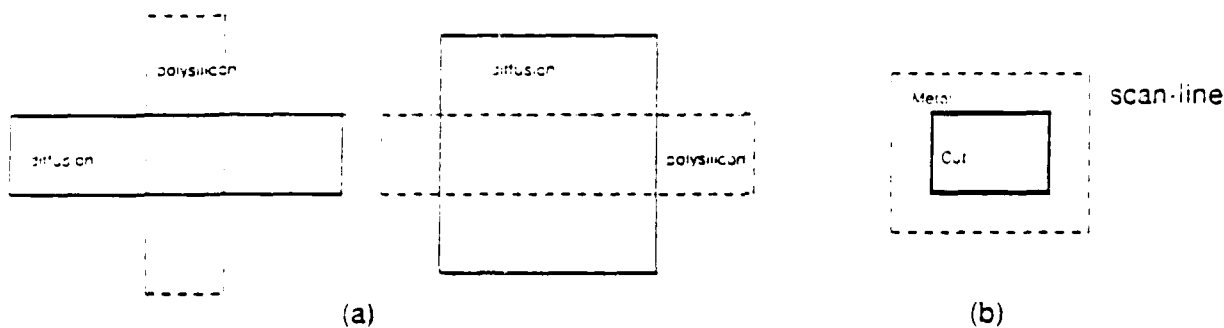
Figure 3-5: Types of intersections

(a) Bilateral intersections, Diffusion ∩ Polysilicon, and (b) unilateral intersection, Cut ∩ Metal.

The distinction between bilateral and unilateral intersection checks has two consequences when writing the connectivity extractor. First, INTERSECT-CHECK must have the ability to detect bilateral intersections from both directions, i.e., when examining either layer type. Secondly, the unilateral intersections of a process dictate the layer ordering for examining mask objects that sort to the same scan-line. If two mask objects which form a unilateral intersection are scanned at the same scan-line, then the object which does not activate the intersection check must be scanned first. Figure 3-6 shows how a contact cut might be missed if this ordering is not observed, and the Metal rectangle is examined last.
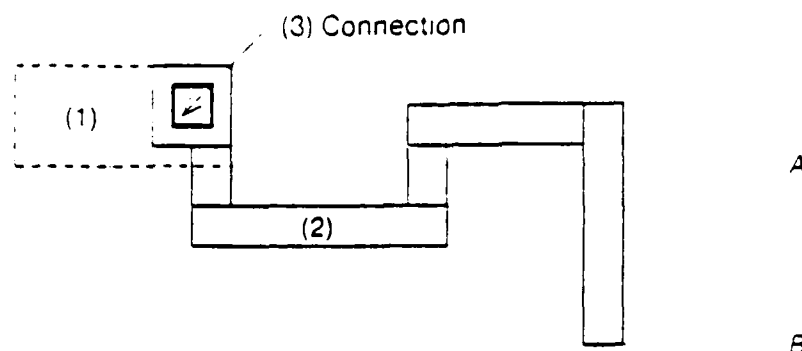
Figure 3-10: Cleaning Network Components

Components (1) and (2) are endpoints of the connection
component, (3). Component (1) can be cleaned when the scan
line passes below A, and (2), when the scan line passes below
B. The connection, (3), cannot be cleaned until after (1) and (2)
are cleaned, as both of these must "verify" the connection,
first.

extracted (figure 3-10). When a network path or switch is extracted, all of its network connections are
verified through the procedure CONTACT-VERIFY.

The EXCL extractor checks all network components, or *cleans* the network, as the scan-line
passes regular y-coordinate intervals. The SCANNER procedure presented in section 3.3.3 calls
NETWORK-CLEAN at a constant y-coordinate interval, *clean-interval*. When a network component is
cleaned, all "extractable" components are (1) removed from the network, N, (2) extracted, and (3)
their circuit element information is written to the circuit file, C.

3.8 Connection of Network

The last subprogram in the sequence of four, FORMAT, has two duties: (1) it connects the
transistor network, and (2) it translates the network into the desired output file format.

```
procedure EXTRACT PATH (path. connection-list. C):
begin
  case path.layer of
        diffusion: c-bottom — c-diff-bottom;
                   c-edge — c-diff-sidewall;
                   j-bottom — j-diff-bottom;
                   j-edge — j-diff-edge;
        polysilicon: c-bottom — c-poly-bottom;
                     c-edge — c-poly-edge;
        metal: c-bottom — c-metal-bottom;
               c-edge — c-metal-sidewall;
        end
  WRITE-NODE(C. path.region);
  area — EX-AREA(path.region);
  perimeter — EX-PERIMETER(path.region);
  capacitance — (c-bottom × area) + (c-edge × perimeter);
  WRITE-GROUND-CAPACITANCE(C. node. capacitance);
  if path.layer = diffusion then
    I — (j-diff-bottom × area) + (j-diff-edge × perimeter);
    WRITE-CURRENT-SOURCE(C. I. region);
    end
  end
```

3.7.2 Network Cleaning

The extractor need not wait until the scanning of all objects has finished before it commences circuit extraction. Often, it is not only possible, but also desirable to convert to circuit elements when possible, for the data storage requirements of a large IC layout are very great, frequently exceeding the primary storage capabilities of a computer system. Once converted to circuit elements, the extractor can store the data until after the completion of scanning; for the EXCL extractor system, when the fourth subprogram starts.

Any path or switch network component can be converted when its complete geometry is known. Due to the nature of the scanning process. a geometric region will not change after the scan line has moved below the minimum y-coordinate of the existing region (figure 3-10). This is, quite simply, because no new mask object could possibly intersect with the region. Checking for this condition is trivial if switch and path minimum y-coordinates are readily available.

Network connections are "extractable" only after the components at both of its endpoints are

```
procedure EXTRACT-PATH (path. connection-list, C):
begin
  case path.layer of
        diffusion: c-bottom — c-diff-bottom;
                   c-edge — c-diff-sidewall;
                   resistivity — rho-diff;
        polysilicon: c-bottom — c-poly-bottom;
                     c-edge — c-poly-edge;
                     resistivity — rho-poly;
        metal: c-bottom — c-metal-bottom;
               c-edge — c-metal-sidewall;
               resistivity — rno-metal;
        end
  new-region-list. new-connection-list. resis-graph —
      EX-RESISTANCE(path.region. connection-list);
  for each new-region. node ∈ new-region-list do
      WRITE-NODE(C. new-region);
      area — EX-AREA(new-region);
      perimeter — EX-PERIMETER(new-region);
      capacitance — (c-bottom X area) + (c-edge X perimeter);
      WRITE-GROUND-CAPACITANCE(C. node. capacitance);
      end
  for each resis-squares. node1. node2 ∈ resis-graph do
      resistance — resis-squares X resistivity;
      WRITE-RESISTANCE(C. node1. node2. resistance);
      end
  for each contact. new-node ∈ new-contact-list do
      CONNECTION-VERIFY(connection. new-node);
      end
  end
```

The second example EX-PATH procedure extracts only the bottom and edge capacitances to ground, and inserts current sources to the substrate for diffusion paths. The current source simulates diffusion leakage current, a parameter which is useful for determining the minimum clock speed of a dynamic charge-storage circuit.

library procedure	parameters	return
EX-AREA	region	area
EX-PERIMETER	region	perimeter
EX-COUPLING	region1, set of other regions, coupling parameters	coupling/edge capacitance matrix
EX-RESISTANCE	region, connection list, minimum resistance	set of new regions, new connection list for each region, resistance network
EX-CONDUCT-DIMENSION[6]	region, connection1, connection2	length, width

Table 3-3: Extraction Library Procedures

---

[6]This procedure is similar to EX-RESISTANCE, but is intended for transistor sizing, for which exact dimensions are important.

3.7 Circuit Element Calculation

During the final stage. the CONNECT subprogram creates the circuit representation. ExCL converts the network components into any number of *circuit elements*.

3.7.1 Circuit Extraction

Circuit extraction is the conversion from network components to a set of equivalent circuit elements following the "extraction model" as described in chapter 2. That is. the circuit extractor begins with geometric data consisting of a set of intersecting rectangles and translates this to a lumped circuit representation containing nodes. capacitors. resistors. etc.

The methods of modelling network components into circuit elements are chosen by the extractor's user. for the modelling methods depend on a number of variables like the IC fabrication technology. the nature of analysis intended for the extracted circuit. and the computation time. A user who wishes to create a new extraction model must write new EXTRACT procedures stating what circuit elements shall be computed. and how. This is done for each network component type: path. switch. and connection.[5] The user is provided with a library of circuit extraction procedures. which greatly simplify his task. Table 3-3 lists the procedures contained in EXTRACT LIBRARY. In the next chapters. these library procedures and their algorithms will be discussed. Note that. where possible. the library procedures return numbers with geometric units like square microns rather than circuit units like Farads. This requires the user to include the final multiplications on circuit values within the EXTRACT procedures. but. it enables him to define which extraction parameters exist. and to supply them with names meaningful for the process.

Below are two example EX-PATH procedures for the NMOS technology. The first procedure extracts resistance and ground capacitance—parameters needed for determining a circuit's maximum speed. When path resistance is computed. the original path is subdivided into any number of paths with a connecting resistor between each. The path geometries are not completely recomputed. only path edge are area values. Underlined and italicized identifiers are user defined "parameters" that can be reset for each invocation of EXCL.

---

[5] Named-points will be skipped in this discussion. for there is generally nothing to do for them. unless the user wishes to include bonding pad parasitic models. the named-point is modelled. simply. as a node.

Connection Type	Endpoint 1 component	Endpoint 2 component
switch node	switch	diffusion path
switch gate	switch	polysilicon path
metal to diffusion contact	metal path	diffusion path
metal to polysilicon contact	metal path	polysilicon path
name	named-point	path
metal to diffusion coupling window[4]	metal path	diffusion path
metal to polysilicon coupling window	metal path	polysilicon path
metal to metal coupling window	metal path	metal path
polysilicon to polysilicon coupling window	polysilicon path	polysilicon path
diffusion to diffusion coupling window	diffusion path	diffusion path

Table 3-2: NMOS connection types

INTERSECT-CHECK creates each network component when the first rectangle is located. Subsequently, it may add new rectangles to a component, subtract existing rectangles from a component (as shown in Sect. 3.3.3), or combine two components of the same type. Component combinations are necessary for regions with two branches which are connected at their lowest points (figure 3-9).
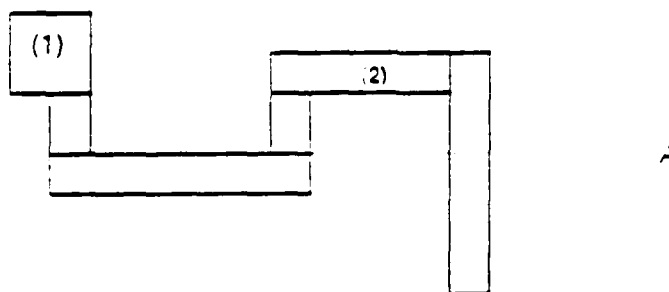
Figure 3-9: Combination of Network Components

While the scan-line is above A. the network contains two components, (1) and (2). When the scanning reaches A, the two are combined.

---

[4] A coupling-window contains paths that are potentially coupled capacitively. See chapter 5

rectangle perimeters minus twice the abutment distance. The non-intersected form has other advantages as shown later.

ITERSECT-REMOVE operates in a y-coordinate scanning method. As INTERSECT-REMOVE moves the horizontal scan position downward, it tacks all x-coordinate positions (or *rays*) contained in the original region. The downward motion of rays "sweeps" out new rectangles of the non-intersected form. When the scan position passes an original rectangle's top or bottom, it rechecks the rays' x-coordinates. Different x-coordinates force INTERSECT-CHECK to make a new horizontal edge in the non-intersected form by finishing the "sweep" of a rectangle and/or by starting a new "sweep" of a rectangle.

3.6 Path, Switch, And Contact Creation

Up to this point, the extraction has proceeded solely on geometric data with geometric operations. After the INTERSECT-CHECK stage, data is grouped into its network *components*. The network contains the complete collection of all components. Initially, there are four types of network components: *paths, switches, connections,* and *named-points.*

A *path* is a continuous region of a conducting layer, uninterrupted by active transistor regions. Metal, polysilicon, and diffusion are the conducting layers of the example NMOS integrated circuit technology.

A *switch* is, naturally, the active region of a transistor. Its definition is, actually, little more than that—the region outlined as the "active" region, a "type" identification, and a list of connection terminals.

A *named-point* in the network viewpoint is unchanged from the layout viewpoint. It is a handle which allows user access to an interesting location of the circuit.

Connections provide the only means of interfacing between the above components. Besides saving its geometric region, a connection provides an identifier for its "type", and two endpoints which may be attached to any path, switch, or named-point. A connection does not always imply an electrical short circuit. As demonstrated in Table 3-2, which lists connection "types" for an NMOS connectivity extractor, a connection might contain a resistive or capacitive component.

1. The P-well regions are scanned first. When the extractor isolates a new region, it adds a watch to $\mathcal{W}$, saving the location of the region. P-well objects are added to the scan-view set, Υ.

2. If other P-well objects intersect with an existing P-well object, additional watches are unnecessary. Regions which merge together as scanning continues (as with the P-well region in figure 3-7) have their watches combined.

3. When contact cut objects are examined, the extractor looks for an intersection between the cut object and a P-well object. If this intersection is found, it searches through Υ, this time for an intersecting P-well region. The watch is removed from a connected P-well region.

4. After the completion of layout scanning, remaining watches are reported, as these watches point to P-well regions which have no ground connection.

3.5 Intersection Removal

Subsequent extraction operations require extensive use of geometric procedures. The extractor is greatly enhanced by the support of an optimized set of general geometric procedures. First, the geometric data representation is converted to a special form of non-overlapping rectangles that allows faster execution of geometric operations. INTERSECT-REMOVE transforms the user-characterized form into a unique non-intersecting form with rectangle abutments only on horizontal edges. Figure 3-8(b) shows the unique *non-intersected* form of the shape originally constructed as in figure 3-8(a). Abutment information is also retained for each rectangle.

(a)   (b)
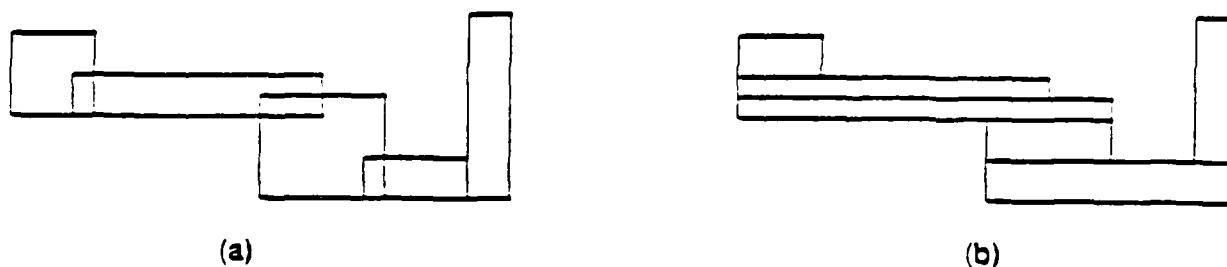
Figure 3-8: Rectangle conversion from random form to non-intersected form

One can easily see that the non-intersected form aids greatly in calculating the region's area and perimeter. The area is the sum of all individual rectangle area, while the perimeter is the sum of all

The INTERSECT CHECK procedure should be easily alterable to accommodate changes in process and extraction modelling. It should be pointed out again that the extractor assumes the input is free of design-rule errors. At this point we can see, however, that a large class of design-rule errors will not disrupt the operation of the extractor: minimum separation rules and minimum overlap rules are in this class. Errors noticed by the extractor are those requiring the existence of a certain layer, the existence of a metal cover over a contact cut, for instance. Violations of these rules should be reported.

3.4.1 Error Watches

Most detectable design-rule errors can be reported immediately, but some cannot, and must take advantage of error *watches*. All detectable errors in the example NMOS process can be reported immediately, but, consider the CMOS rule stated below and shown in figure 3-7:

> All P-wells must be connected to ground through a region of P^+ diffusion which is connected to GND metal through a contact cut.
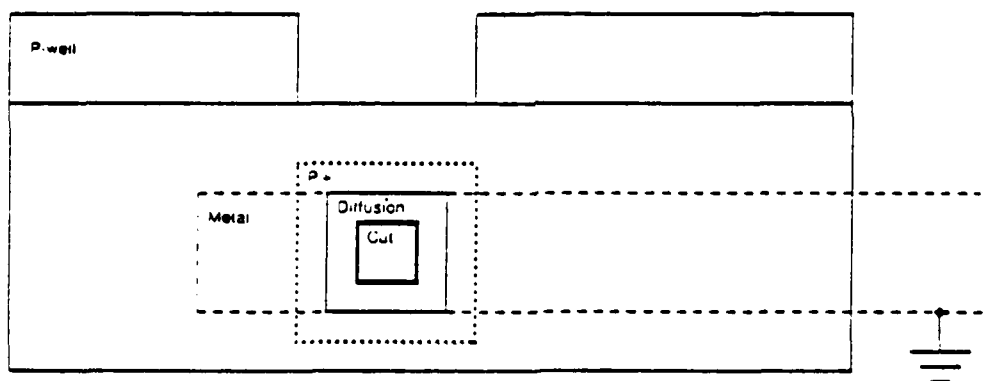
Figure 3-7: CMOS rule for error watches

The extractor must verify that within each P-well region, there exists at least one location where both the P^+ and cut layers also exist. The extractor cannot perform the error check when either the cut or P^+ objects are scanned, for these are used differently at other locations. In general, watches are useful when extraction checks are made in a region for the occurrence of another object within it.

The following sequence of events takes place in INTERSECT-CHECK to locate violations of the CMOS rule.

43

```
named-point:
    begin
        {make connection to appropriate layer}
        case □r.point-layer of
            diffusion : for each Diffusion intersection. □d, and its path, p. in Q do
                    NAME-NETWORK-PATH(N, p, □r.point-name);
                    end
            polysilicon : for each Polysilicon intersection. □p. and its path, p. in Q do
                    NAME-NETWORK-PATH(N, p, □r.point-name);
                    end
            metal : for each Metal intersection. □m, and its path, p, in Q do
                    NAME-NETWORK-PATH(N, p, □r.point-name);
                    end
            end
        end
    end
return
end
```

```
cut:
    begin
        {find intersecting metal rectangle}
            for each Metal intersection, □m, and its path. p, in Q do
                top-path ← p;
                end
        poly-found ← false;
        {search for polysilicon intersection first}
        for each Polysilicon intersection, □p, and its path. p, in Q do
            bottom-path ← p;
            poly-found ← true;
            □cut-rest ← □r ∩ □p̄ ;
            if REGION-EXISTS(□cut-rest) then
                QUEUE-ADD(□cut-rest);
                end
            end
        if ~poly-found then
                {otherwise locate intersecting diffusion rectangle}
                    for each Diffusion intersection, □d, and its path. p, in Q do
                        bottom-path ← p;
                        □cut-rest ← □r ∩ □p̄ ;
                        if REGION-EXISTS(□cut-rest) then
                            QUEUE-ADD(□cut-rest);
                            end
                        end
            end
        {make connection}
        CONTACT-NETWORK-PATHS(N, top-path, bottom-path);
        end
```

<u>channel</u>:
    begin
        {determine transistor type by implant mask}
        if $\Box r$ intersects any Implant in Q
          then *switch-type* ← <u>depletion</u>
          else *switch-type* ← <u>enhancement</u>
          end
        switch ← NEW-NETWORK-SWITCH($\mathcal{N}$, $\Box r$, *switch-type*);
        {locate source or drain intersections}
        for each Diffusion intersection. $\Box d$, and its path, p, in Q **do**
          ADD-SOURCE/DRAIN($\mathcal{N}$, s, p);
          end
        {locate other transistor rectangles}
        for each Channel intersection. $\Box c$, and its switch, s, in Q **do**
          COMBINE-NETWORK-SWITCHES($\mathcal{N}$, s, *switch*);
          end
        {locate transistor gate}
        for each Polysilicon intersection, $\Box p$, and its path, p, in Q **do**
          ADD-GATE($\mathcal{N}$, s, p);
          end
        SCAN-VIEW-ADD(Q, $\Box r$, *switch*);
        end

<u>implant</u>:
    begin
        {store in $\mathcal{V}$}
        SCAN-VIEW-ADD(Q, $\Box r$, --null--);
        end

<u>metal</u>:
    begin
        metal-path ← NEW-NETWORK-PATH($\mathcal{N}$, $\Box r$);
        {locate other path rectangles}
        for each Metal intersection, $\Box m$, and its path, p, in Q **do**
          COMBINE-NETWORK-PATHS($\mathcal{N}$, p, *metal-path*);
          end
        SCAN-VIEW-ADD(Q, $\Box r$, *metal-path*);
        end

{INTERSECT-CHECK for NMOS process. Inter-nodal capacitance windows are not located.}
procedure INTERSECT-CHECK ($\Box r$, t, Q, W, N):
begin
  case $\Box r$.*layer* of
    <u>diffusion</u>:
        begin
            {locate transistor regions}
            for each Polysilicon intersection, $\Box p$, in Q do
                $\Box$*channel* — $\Box r \cap \Box p$;
                QUEUE-ADD(Q, $\Box$*channel*);
                $\Box$*diffusion-fragments* — $\Box d \cap \overline{\Box channel}$;
                QUEUE-ADD(Q, $\Box$*diffusion-fragments*);
                return;
                end
            diff-path — NEW-NETWORK-PATH(N, $\Box r$);
            {locate other path rectangles}
            for each Diffusion intersection, $\Box d$, and its path, p, in Q do
                COMBINE-NETWORK-PATHS(N, p, *diff-path*);
                end
            {locate source or drain intersections}
            for each Channel intersection, $\Box c$, and its switch, s, in Q do
                ADD-SOURCE/DRAIN(N, s, *diff-path*);
                end
            SCAN-VIEW-ADD(Q, $\Box r$, *diff-path*);
        end

    <u>polysilicon</u>:
        begin
            poly-path — NEW-NETWORK-PATH(N, $\Box r$);
            for each Diffusion intersection, $\Box d$, and its path, p, in Q do
                REMOVE-FROM-PATH(N, p, $\Box d$);
                {locate transistors}
                $\Box$*channel* — $\Box r \cap \Box d$;
                QUEUE-ADD(Q, $\Box$*channel*);
                $\Box$*upper-d-fragment*, $\Box$*lower-d-fragments* — $\Box d \cap \overline{\Box channel}$;
                ADD-TO-PATH(p, $\Box$*upper-d-fragment*);
                QUEUE-ADD(Q, $\Box$*lower-d-fragments*);
                end
            {locate other path rectangles}
             for each Polysilicon intersection, $\Box p$, and its path, p, in Q do
                COMBINE-NETWORK-PATHS(N, p, *poly-path*);
                end
            SCAN-VIEW-ADD(Q, $\Box r$, *poly-path*);
        end

(a)

(b)   (c)   (d)
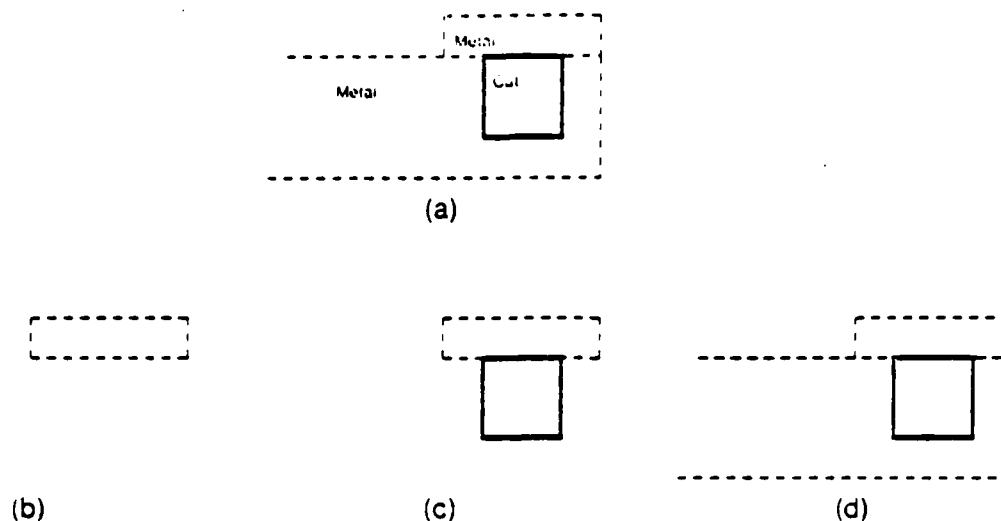
Figure 3-6: A missed Metal-Cut intersection

(a) Original layout. The upper metal rectangle is examined (b).
Then, when the scan-line moves down to the other boxes, the
cut rectangle is examined next (c), and no intersection with
metal is found. When the metal rectangle is examined, no
intersection check with cut is made.

The complete INTERSECT-CHECK algorithm for the NMOS process is shown below. Procedures involving the network, N, will be discussed later. For clarity, variable names preceded by a box, $\square$, refer to mask-object variables. Constant identifiers are underlined.

3.8.1 Minimum Circuit Values

We generally wish the output network to have only those circuit elements whose values are above a certain minimum threshold. Circuit elements below the threshold are deemed as having trivial effects on circuit performance, and thus, should be removed. They are likely to add to analysis (or simulation) time with little benefit.

The minimum circuit values are set by user defined parameters supplied to the network connection subprogram. These need not match the minimum values of CONNECT's EXTRACT, and the user may wish to alter these if the desired circuit analysis tolerances change. He can also override the minimum circuit values for a specific node by declaring it *critical*. Circuit elements connected to a critical node are never ignored.

The network connection subprogram may set some of the minimum values on its own if the output format type dictates this. An output format for switch level simulation cannot, in general, have interconnect resistors. In this situation, the minimum interconnect resistance is infinite.

3.8.2 Connection

The network produced by EXCL contains "path nodes", "switches", and "named-points" attached by "connections". Many connections are short-circuits. Figure 3-12 depicts the network extraction of the layout shown in figure 3-11. The dark lines in the network figure represent connections. During the connection stage of the fourth subprogram, nodes which are connected through short-circuit connections or through resistors with values below the minimum threshold are eliminated. Figure 3-13 shows the resulting network where:

$$R_{d1}, R_{d3}, R_{d4}, R_{contact} < R_{minimum}, \text{ and}$$
$$R_{d2}, R_{p1}, R_{p2} \geq R_{minimum}.$$

Figure 3-14 shows the resulting network if all resistors are below the minimum, $R_{minimum}$.
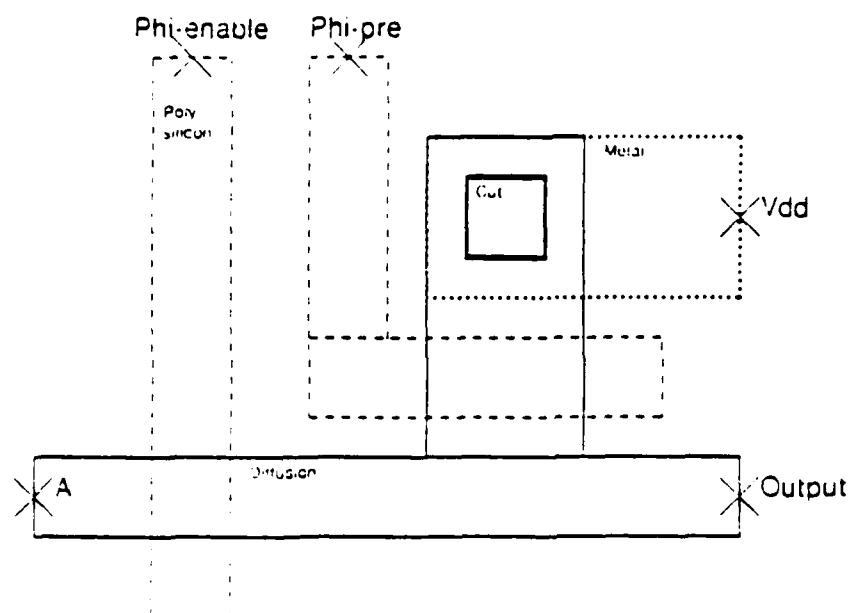
Figure 3-11: Example Layout

3.9 Output Format

The final stage. after network connection. is output formatting. A different program module is required for each output format. Available formats are the following:

1. a source file for the circuit simulator (SPICE [9]),

2. a source file for the switch-level simulator (MOSSIM [7] or NL/RNL [8])

3. a tabular format readable by humans who are interested in scanning the nodal capacitances or resistances of their layout or who are interested in locating internal node names, and

4. a node list, readable by the layout editor for displaying node names, and other information on the nodes.
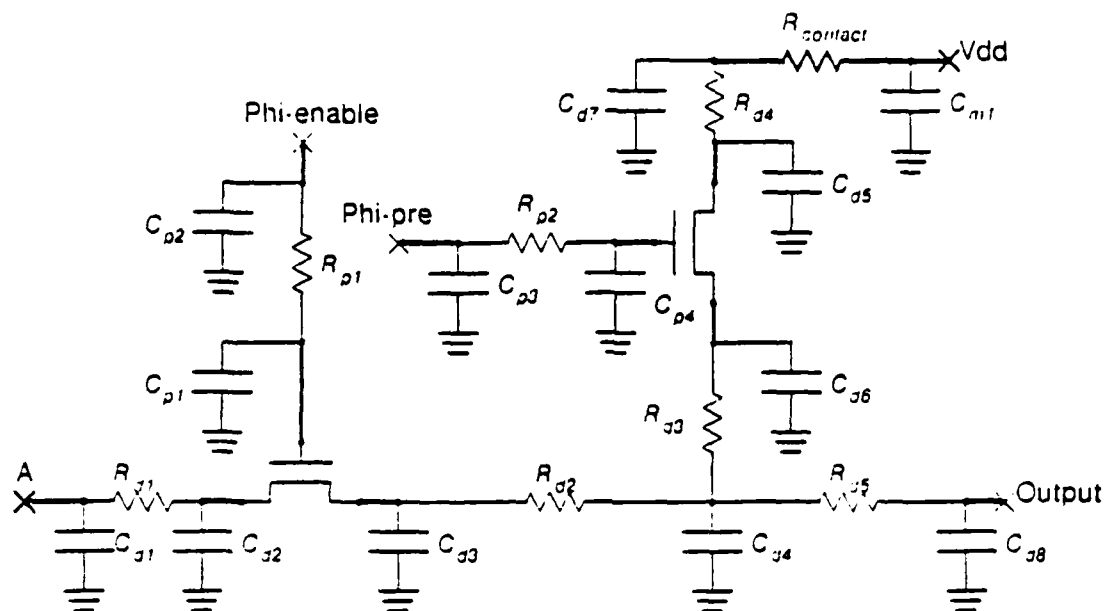
Figure 3-12: Complete Extracted Network of Example Layout

"Connections" are shown as darker lines. All "connections" are short-circuits except the contact cut, which has an associated resistance.
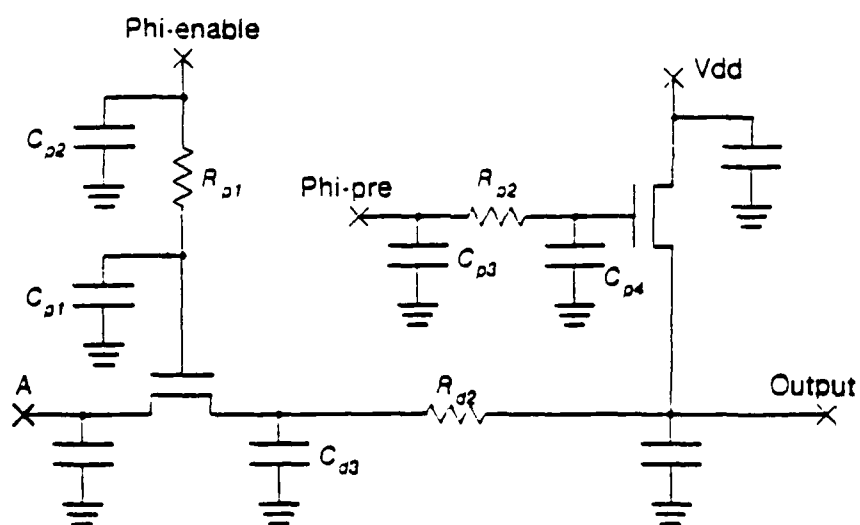
Figure 3-13: Connected network

54
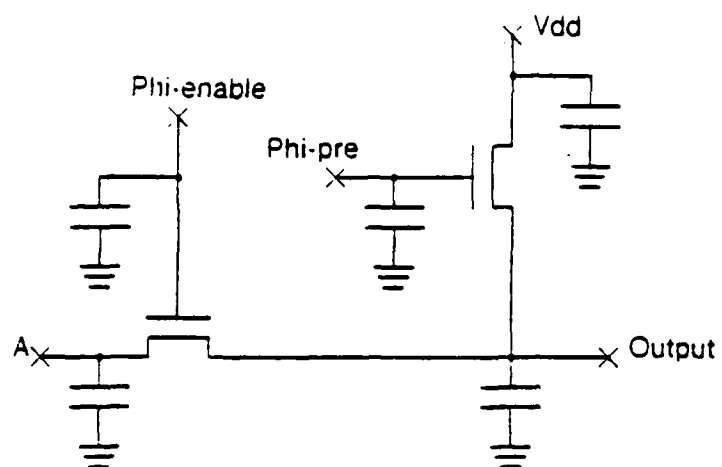
Figure 3-14: Connected network with all resistors ignored

55

CHAPTER FOUR

Resistance Extraction

The last chapter covered algorithms for examining geometric regions in an IC mask layout, for isolating and grouping geometries to model into lumped circuit elements, and lastly, for generating a usable and informative network description from the extracted circuit elements. In this chapter, we will begin to look at the algorithms that fit between the geometric stage and the circuit stage— algorithms that extract lumped circuit equivalents from IC geometries. The algorithms of the last chapter are sufficient to write a "connectivity extractor"; this chapter and the next two contain information on additional algorithms needed to write a "circuit extractor".

This chapter begins the discussion of circuit extraction by explaining the library procedure EX-RESISTANCE listed in table 3-3. The resistance extractor is given a geometric region and a list of two or more "connections". Its task is to calculate a lumped resistor network between the connections that models the voltage-current relationship of a planar resistive conductor of the given geometry.

The resistance extractor initially divides the geometric region into subregions. Then it calculates an equivalent resistive subnetwork for each subregion. Lastly, resistance subnetworks are combined into a complete resistive network. Dividing geometric regions into subregions follows one restriction: dividing lines must be along equipotential lines. The reasons for this will become clear in later sections.

First, we will discuss the general field analysis for resistive regions, followed by the three algorithms used by the extractor for resistance calculation of subregions.

4.1 Field Analysis of Electric Conduction

The two fields of interest in solving the problem of electric conduction are the electric potential, V, and current density, $\vec{J}$. The electric field, defined by

$$\vec{E} = -\nabla V \tag{4.1}$$

is sometimes more useful than V. The relationship between the two fields is governed by Ohm's Law,

$$\vec{J} = \sigma \vec{E}, \tag{4.2}$$

where σ is the conductivity or the inverse of resistivity, $1/\rho$.

The physical area of the problem is divided into three regions:

1. the conductor region, $\mathcal{C}$, where V is constant, or $\sigma = \infty$ (a connection from chapter 3),

2. the resistive region, $\mathcal{R}$, where σ has a constant, non-zero value (a path from chapter 3), and

3. the insulator region, $\mathcal{I}$, where $\sigma = 0$ (everywhere else).

The boundary between $\mathcal{C}$ and $\mathcal{R}$ forms the *principle boundary*, $\mathcal{P}$, and follows the Dirichlet boundary condition,

$$V(P) = V_C. \tag{4.3}$$

The *natural boundary* between regions $\mathcal{R}$ and $\mathcal{I}$ follows the Neumann boundary condition,

$$\nabla V \cdot \vec{n} = 0, \tag{4.4}$$

where n is a normal to the boundary $\mathcal{P}$.

The analysis of a resistive region begins by forcing a known voltage difference, V_d, between two conductor regions. The potential, $V(x, y, z)$, is found in the resistive region with the aid of Laplace's equation,

$$\nabla^2 V = 0, \tag{4.5}$$

which holds everywhere in $\mathcal{R}$. Then the total current flowing into one conductor region, $\mathcal{C}$, is found through Gauss' Divergence Theorem by summing the current density flowing over $\mathcal{P}$, a surface at the boundary between $\mathcal{C}$, and $\mathcal{R}$. That is:

$$I_d = \int_{P_l} \vec{J} \cdot \vec{n} \, dP.$$

where the vector $\vec{n}$ is a normal to the surface P_l pointing into C_l. Substituting equations (4.1) and (4.2) into the last gives

$$I_d = \sigma \int_{P_l} \vec{E} \cdot \vec{n} \, dP = -\sigma \int_{P_l} \nabla V \cdot \vec{n} \, dP. \tag{4.6}$$

The equivalent lumped resistance over the whole region is then calculated with

$$R_d = \frac{V_d}{I_d}. \tag{4.7}$$

Regions with more that two principle boundaries require more than one voltage setting and current integration. More on this will be presented later.

4.2 Finite Elements Techniques

Finite element techniques are well suited to extracting resistance networks of any geometric shape with any number of external connections or principle boundaries.

In general, finite element analysis involves subdividing a continuous region into a large number of small regions or *finite elements*. One or more unknown field values are calculated for each finite element, thereby approximating the fields over the whole region. Finite element methods often succeed with computer analysis. because the computer can calculate the unknowns for many small finite elements with simpler models that are independent of overall shape than it can calculate unknowns for a general continuous region.

For the finite element analysis of planar resistance, we divide the resistive region, $\mathcal{R}$, into equally sized finite elements across which we calculate the fields for voltage and current density. We can safely assume that the components of current density, $\vec{J}$, are limited to the x and y directions—parallel to the IC surface—thereby eliminating the need for separate finite elements in the z direction.

Although a few two-dimensional grid topologies are feasible , the natural choice for us is the square grid. due to the orthogonal shapes of IC conductor geometries. Figure 4-1 shows a section of $\mathcal{R}$ divided into finite elements. In the center of each we place a node. Node i shown in figure 4-2 has four nearest neighboring nodes in $\mathcal{R}$. denoted j, k, l, and m—each a distance a from i. The node voltages are V_i, V_j, V_k, V_l, and V_m, respectively.
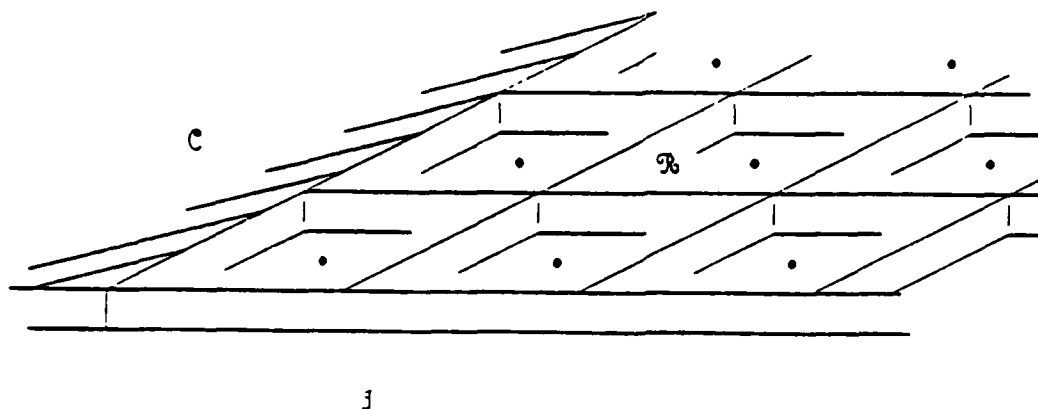
Figure 4-1: Finite elements of a planar resistor
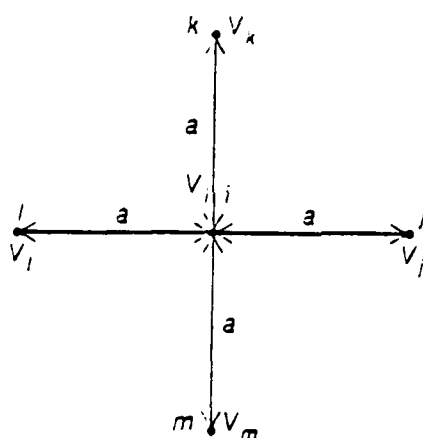
Figure 4-2: Node designations for node i and its neighbors

We now wish to develop difference equations from Laplace's equation (equation (4.5)) for the discrete point at node i. We begin by expanding $V(x)$ into a Taylor's Series about point i, and evaluating it at point j. This gives

$$V_j = v + a\left(\frac{\partial V}{\partial x}\right)_i + \frac{a^2}{2!}\left(\frac{\partial^2 V}{\partial x^2}\right)_i + \frac{a^3}{3!}\left(\frac{\partial^3 V}{\partial x^3}\right)_i + \frac{a^4}{4!}\left(\frac{\partial^4 V}{\partial x^4}\right)_i + \ldots \qquad (4.8)$$

Similarly, the Taylor's series evaluated at l yields

59

$$V_l = v_i - a\left(\frac{\partial V}{\partial x}\right)_i + \frac{a^2}{2!}\left(\frac{\partial^2 V}{\partial x^2}\right)_i - \frac{a^3}{3!}\cdot\left(\frac{\partial^3 V}{\partial x^3}\right)_i + \frac{a^4}{4!}\left(\frac{\partial^4 V}{\partial x^4}\right)_i - \ldots \tag{4.9}$$

Adding the above two equations and ignoring terms of fourth order or more gives

$$V_j + V_l = 2V_i + \frac{2a^2}{2!}\left(\frac{\partial^2 V}{\partial x^2}\right)_i .$$

or

$$a^2\left(\frac{\partial^2 V}{\partial x^2}\right)_i = V_j + V_l - 2V_i . \tag{4.10}$$

Similarly, for the y-direction:

$$a^2\left(\frac{\partial^2 V}{\partial y^2}\right)_i = V_k + V_m - 2V_i . \tag{4.11}$$

Adding (4.10) and (4.11),

$$a^2\left(\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2}\right)_i = V_j + V_k + V_l + V_m - 4V_i . \tag{4.12}$$

For Laplace's equation to hold at node i, the left hand side of (4.12) is zero,[7] or

$$V_j + V_k + V_l + V_m - 4V_i = 0. \tag{4.13}$$

Stated simply, the difference equation, (4.13), says that V_i is the average of the voltages on each of its nearest neighboring nodes.

We can extend our view of the difference equation to analogous discrete resistor networks. Consider the circuit of figure 4-4. Kirchhoff's current law at node i states

$$\frac{V_j - V_i}{R} + \frac{V_k - V_i}{R} + \frac{V_l - V_i}{R} + \frac{V_m - V_i}{R} = 0,$$

or

$$V_j + V_k + V_l + V_m - 4V_i = 0 . \tag{4.14}$$

This is identical to (4.13). Thus, we see that solving the finite element analysis for the V and $\vec{J}$ fields is analogous to solving the network shown in figure 4-4. All resistors between nodes within $\Re$ have the value R equal to ρ_{sn}, the resistivity of one square in $\Re$.

---

[7] Note also that for Laplace's equation to hold, the higher order derivatives that we ignored in equation (4.10) are also zero.
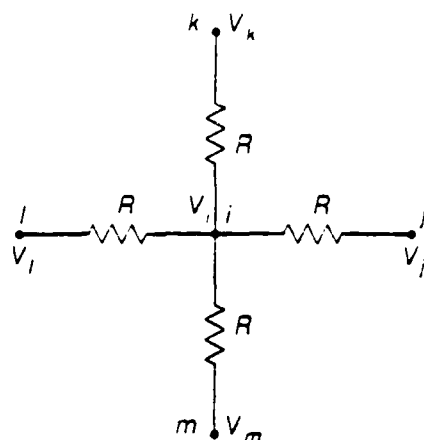
Figure 4-3: Resistor network analogy of a single finite element
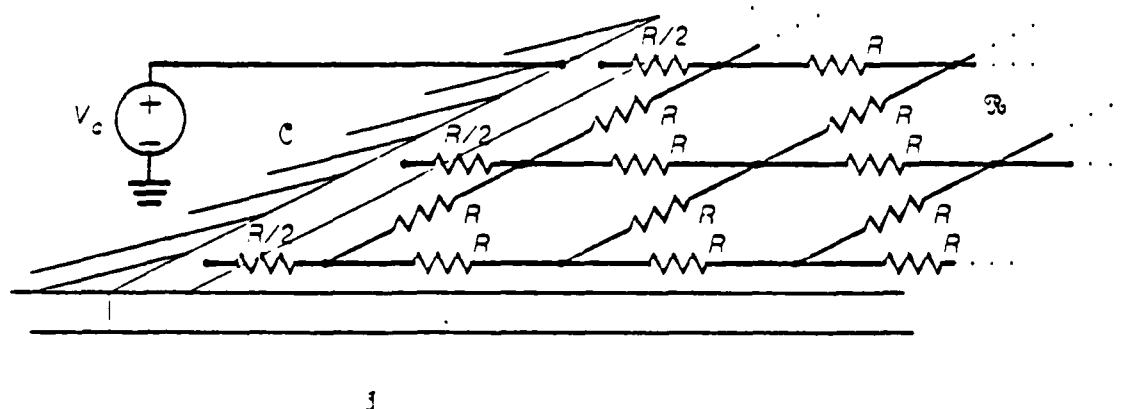
Figure 4-4: Finite element resistor network

4.2.1 Boundary Conditions

We will complete the fields/network analogy by examining the behavior at the two boundary types. At the principle boundary between C and $\mathfrak{B}$, the Dirichlet condition of equation (4.3) simply states that the conductor voltage is known and invariant. We can, therefore, model the conductor boundary as a series of nodes connected to an independent voltage source, V_c. A resistor connects the boundary nodes with the nodes in $\mathfrak{B}$ as shown in figure 4-4. The resistor's value is $R/2$, however,

since the distance between the conductor and the nearest node in $\mathcal{R}$ is $a/2$.[8]

The Neumann boundary condition (equation (4.4)) reduces to the obvious fact that no current flows across the natural boundary between $\mathcal{R}$ and $\mathcal{J}$. The network has no resistors at the natural boundary.

4.2.2 Direct Solution of V

Both a *direct* technique and an *indirect* or *iterative* technique are presented for solving the voltage distribution across $\mathcal{R}$. Since the solution of V requires a large portion of the computation time, different approaches are used in an effort to optimize computer time.
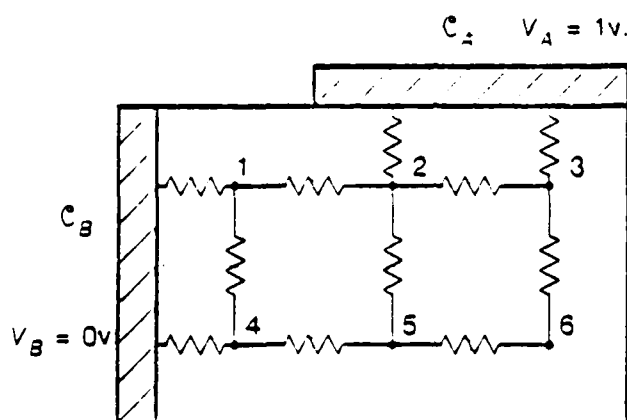
Figure 4-5: Network for direct solution example

In this section we will use the network analogy for the region shown in figure 4-5 to visualize the direct matrix solution of V. Initially, known voltages, V_A and V_B are placed on the conductor regions. A node equation similar to equation (4.14) is written for each of the six nodes in $\mathcal{R}$. In matrix form these equations are

---

[8] Although the choice of $a/2$ seems fairly obvious, the approximation may be derived by starting with equation (4 8), substituting a with $a/2$, and adding it twice to equation (4 9). We also must approximate that $\dfrac{\partial^2 V_r}{\partial r^2} = 0$ near the boundary, where V_r is the voltage tangential to the boundary.

$$\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 \\ -1 & 5 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 \\ -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

or

$$K \cdot v = u. \tag{4.15}$$

Note that the extractor can determine the K and u matrices directly by inspecting the network. For each node, i, in the network, we fill in the ith row of K and u. In K a (-1) is placed in the jth column if a resistor connects nodes i and j. The diagonal term, K_{ii}, equals the total number of resistors connecting to node i, with resistors to C regions counting double. The ith term of u is $2 \times V_C$ for each resistor connecting i to C.

To solve for v in equation (4.15) we may use any of the standard matrix algebra techniques. Band matrix techniques have been developed and studied for systems of linear equations like (4.15) that arise from finite element analysis. The techniques are introduced in Appendix B and are discussed more thoughly by Whiteman [15] and Wilkinson [16].

4.2.3 Iterative Solution of V

Unlike the direct solution, the *indirect* or *iterative* solution does not find the exact solution, but rather, it successively approximates the solution to v. On each iteration the approximation, $\hat{v}$, becomes closer to the true solution. Iteration halts when the difference between $\hat{v}$ and v reaches the desired accuracy.

Equation (4.13) is valid at each node only for the exact solution, but is not in general valid for $\hat{v}$. The difference due to approximation—called the *residual*—is

$$\psi_i = \hat{v}_j + \hat{v}_k + \hat{v}_l + \hat{v}_m - 4\hat{v}_i.$$

During each iteration, the new value of $\hat{v}_i$ is calculated such that the residual is zero, or.

$$\hat{v}_i^{n+1} = \frac{1}{4} (\hat{v}_i^n - \hat{v}_k^{n+1} + \hat{v}_l^{n+1} + \hat{v}_m^n) \tag{4.16}$$

where the superscript refers to the iteration number. The newer $n+1$ iteration terms are used for neighbors k and l (assuming nodes are scanned from left to right and from top to bottom); values at j and m have not yet been calculated. The residual, ψ_i, does not remain zero, for $\hat{v}_j$ and $\hat{v}_m$ change too, but, overall, the residuals decrease on each iteration.

We cannot know exactly how much the estimates, $\hat{v}$, differ from the true values, v, for v is not known. However, Milne has shown [17] that an upper bound on accuracy is given by

$$error \leq \frac{\psi_{max} \, \rho^2}{4} \tag{4.17}$$

where ψ_{max} is the maximum residual over all the nodes, and ρ is the maximum radius of a circle enclosing ℛ, measured in inter-nodal spacings.

The rate at which $\hat{v}$ converges to v is increased if we "over-estimate" the new values for $\hat{v}$. This technique know as *successive over-relaxation* was discovered in the hand calculation days. The new estimates are calculated by

$$\hat{v}_i^{n+1} = \hat{v}_i^n + \frac{\omega}{4} \, \psi_i = \hat{v}_i^n + \omega \left(\frac{\hat{v}_j^n + \hat{v}_k^{n+1} + \hat{v}_l^{n+1} + \hat{v}_m^n}{4} - \hat{v}_i^n \right). \tag{4.18}$$

The *overrelaxation factor*, ω, effects the rate of convergence and is between 1 and 2. For $\omega = 1$, equations (4.18) reduces to (4.16), and for $\omega \geq 2$, the iteration becomes unstable or diverges.

Although much investigation has been done into finding the optimum overrelaxation factor, ω_{opt}, based on a region's shape or size, Forsythe and Wasow [18] have derived a reasonable alternative. Their value for the optimum overrelaxation factor is

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \lambda}} ,$$

$$\text{where } \lambda = \lim_{n \to \infty} \frac{\psi_{max}^{n+1}}{\psi_{max}^n} .$$

The value for λ cannot be calculated exactly, but is estimated at even intervals of n as n increases.

4.2.4 Comparison of the Direct and Iterative Solutions

The selection between the direct and iterative methods for finding v is based on computation time and computer memory requirements. For the direct solution, no algorithm has been found superior to the *Gaussian elimination* method. For n finite elements the general Gaussian elimination method requires $\frac{n^3}{3}$ operations (adds, multiplies and divides) during the primary workload, matrix triangularization. n^2 memory locations are needed. Refinements on Gaussian elimination for band-limited matrices reduce the complexity significantly. (See Appendix B.) If the matrix's maximum bandwidth is s (which equals the maximum number of elements on any row), then the operation count reduces to approximately ns^2 while the storage requirement reduces to ns.

In contrast, the iterative solution needs n memory locations and approximately $6Kn$ operations. The number of iterations, K, is difficult to characterize, for it is a function of n, the shape of $\mathcal{B}$, and the accuracy. Figure 4-6 shows a graph of K vs. $\sqrt{n}$ for a fixed shape and an estimated accuracy of 2% based on the criterion of equation (4 17). The number of iterations is approximately proportional to the number of elements along a line, or $K \propto \sqrt{n}$.

Combining all of the above effects we see that operation count for the direct and indirect methods are proportional to n^2 and $n^{3/2}$ respectively. Typically, the band-matrix Gaussian elimination method is better for solutions with small n. For storage and computation considerations, we must rely on successive overrelaxation methods when n exceeds a threshold (around 1500 for EXCL).

4.2.5 Calculation of R_d and Resistor Networks

Once the voltage distribution is known, we find the current between conductor regions as prescribed by equation (4.6). The current calculation surface lies at the boundary formed between C and $\mathcal{B}$. To calculate the current flowing into the C region of figure 4-4, we sum all currents flowing through resistors with value $R/2$. Translating equation (4.6) to fit the discrete case, we get

$$I_d(C_1) = \frac{2}{R} \sum_{\substack{i \in \text{all nodes} \\ \text{adjacent to } C_1}} (v_i - v_{C_1}).$$

Finding the lumped equivalent resistance, R_d, is trivial (equation (4.7)).

We now consider the question of how to compute a resistor network via finite element techniques for a region containing more than two principle boundaries. The resistive region is

CHAPTER FIVE

Interconnection Capacitance Extraction

The two problems in extracting capacitance networks of interconnecting lines differ vastly. One problem, that of extracting inter-nodal capacitance, presents the greatest extraction challenge to solve completely and accurately. Shortcuts can reduce the problem but invariably lead to inaccurate modelling in certain cases. The other problem—extracting ground capacitance between a conductor and the substrate—could be called trivial, were it not for its slight dependence on inter-nodal capacitance. Chapter 2 gave definitions for each of the capacitances of interest—*inter-nodal*, *edge*, and *bottom* capacitance. They are re-illustrated here in figure 5-1. *Ground capacitance* refers to the sum of edge and bottom capacitances, as both of these connect to the substrate.
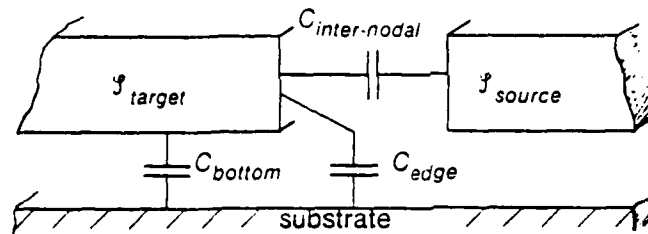
Figure 5-1: Capacitance types for a multi conductor system.

This chapter, first, covers algorithms for inter-nodal capacitance extraction, and then covers algorithms for ground capacitance extraction.
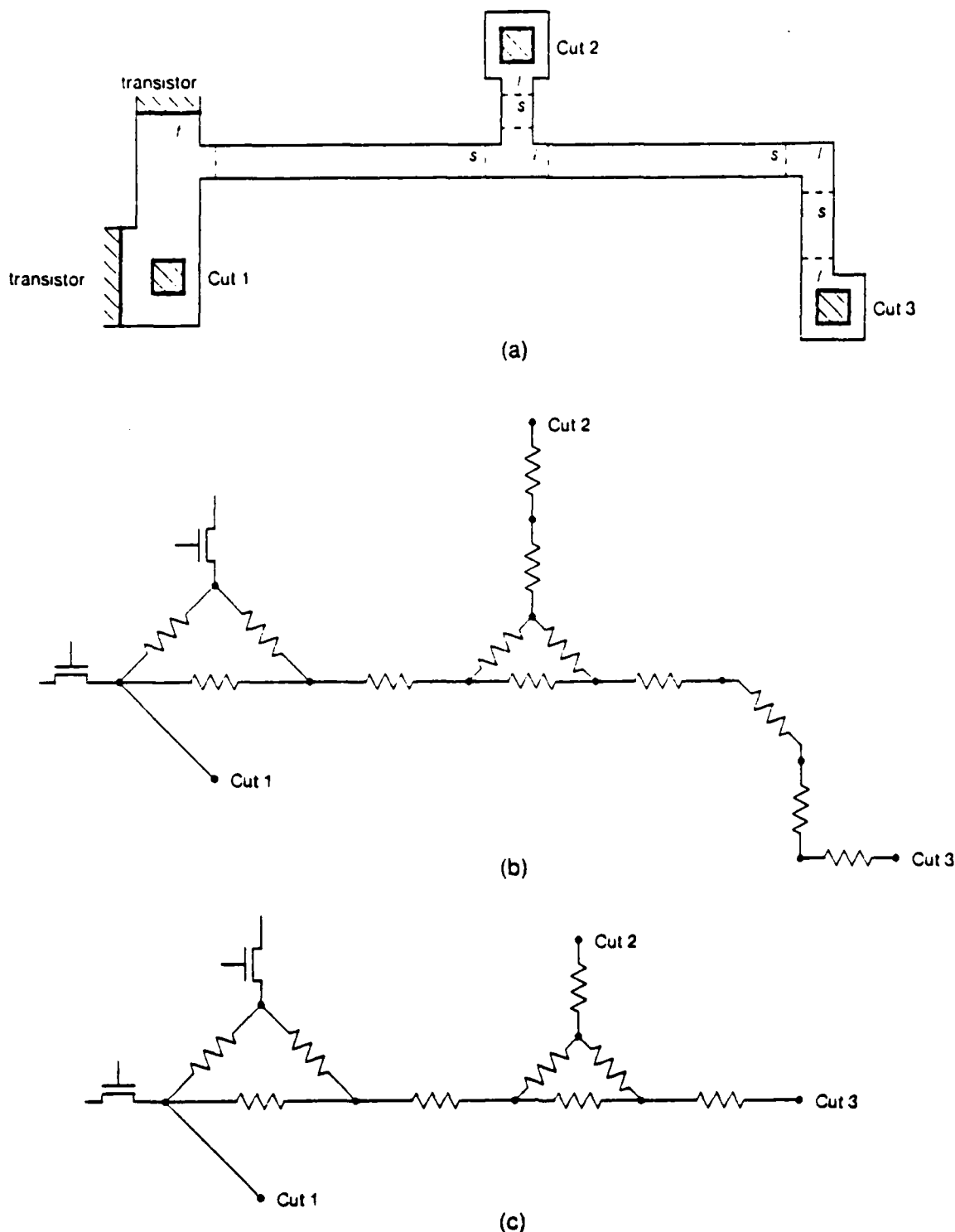
Figure 4-14: Resistance extraction of complete region

(a) original resistive region divided into subregions. Subregions are marked with extraction method: f = finite element analysis, s = straight resistor, and l = library lookup. (b) subnetworks of each subregion. The distance between Cut 1 and the lower transistor is less than d_w. (c) final network.

4.7 Geometric Information for Capacitance Extraction

In the next chapter we will find that capacitance extraction, which follows resistance extraction, needs two forms of geometric information from the resistance extractor to properly model distributed *RC* interconnections.

- It needs quantified area and perimeter estimates for each node of the resistance network (for ground capacitance extraction).
- The inter-nodal capacitance extractor needs the entire geometric description of the conductor, along with the placement of internal and external "connection" regions.

The second requirement is easily fulfilled by returning the original geometric information with the addition of subregion "connections".

It is fortunate that substrate capacitance extraction does not need actual geometries, but only numerical values for area and perimeter of each new node. To return the first requirement, the area and perimeter of each subregion is divided equally among all "connections" to that subregion. Since each node of the final network represents one or more internal and/or external connections, the values returned comprise the sum of all area and perimeter contributions from its connections.

Figure 4-14 shows for a sample IC conductor the resistor subregions, their subnetworks, and the final network.

pointing to the next clue. The search process traverses the tree structure, checking for valid answers to each clue. If the shape has valid answers to all clues, the search is successful, and the leaf node contains the equivalent resistor subnetwork. The tree hierarchy follows the clues listed below:

- the number of rectangles. This node immediately drops a number of complex shapes from the search.
- the number of network connections.
- the upper left and lower right coordinates of the first rectangle. The origin is always defined as the upper left of the first rectangle, and to retain scale independence, all coordinates are normalized to a unit defined by $1/256$ of the first rectangle's height. This clue is repeated for each rectangle.
- the upper left and lower right coordinates of the first boundary. The same coordinate and scale transformations apply as before. This clue is repeated for each connection.

To complete this section, it should be noted that any subregion failing to match a library shape is rendered "irregular" and must submit to finite element analysis.

4.6 Connecting Resistor Subnetworks

After all subregions are converted to resistor subnetworks, the extractor connects the subnetworks to form the complete resistor network. All "connections" are transformed into nodes. *External nodes* are transformations of "connections" <u>provided to</u> the resistance extractor. *Internal nodes* may form at borders between adjacent subregions External nodes must remain, however, one can simplify the network structure if simplification removes only internal nodes. Generally, only series resistors are found in IC interconnection modelling. The extractor recognizes series resistors and deletes the internal node.

Not all series resistors are combined, for this may destroy the distributed RC modelling. As pointed out in section 2.5 a very long distributed RC line is better modelled with a finer resolution of discrete nodes. Series resistors are combined only if the combined resistance is less than the maximum resistance. In fact, if a straight resistor exceeds the maximum resistance value even before network optimization, then <u>extra</u> internal nodes are added.

76
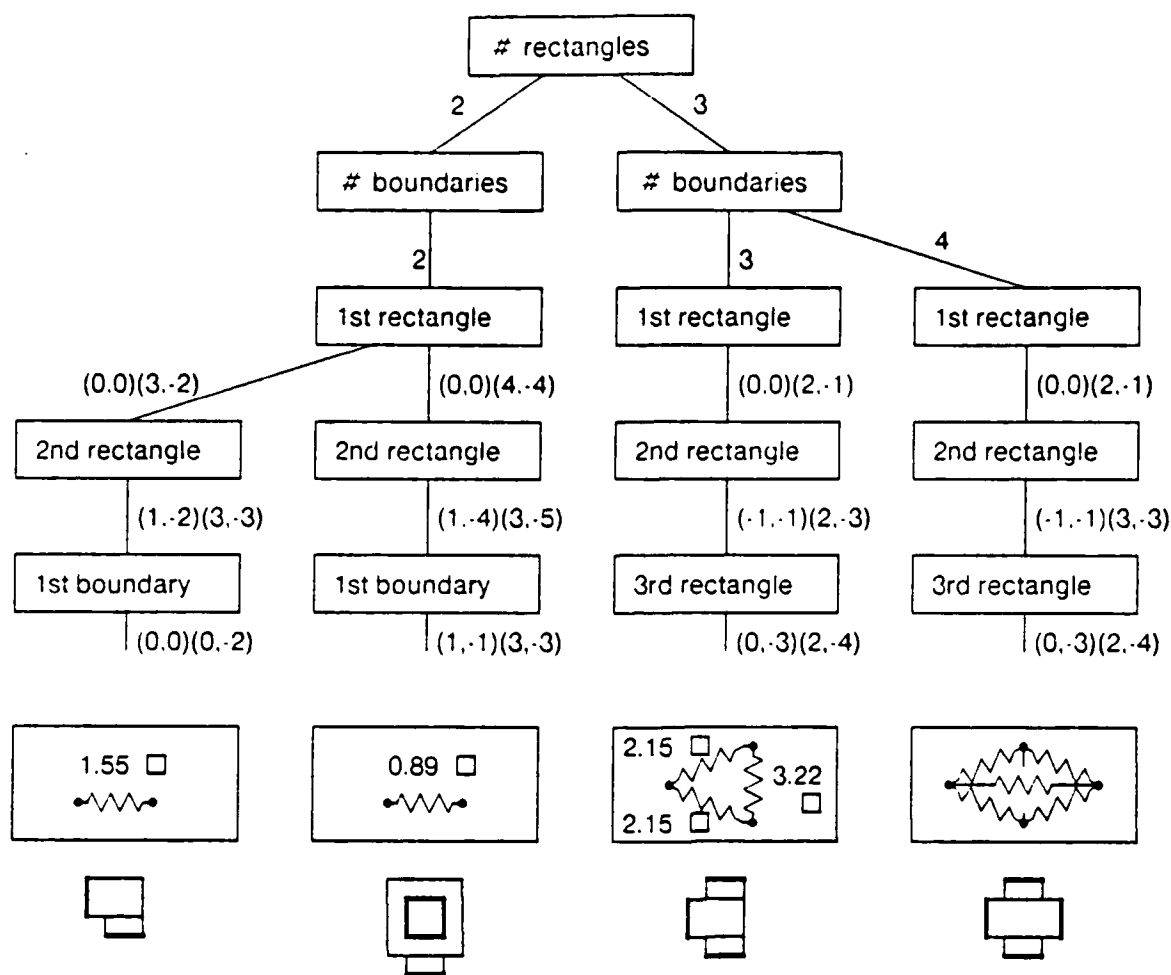
Figure 4-13: Library tree structure for four entries

Resistor shapes are shown below each corresponding entry.
For simplicity, an absolute scale has been used.

An incredibly powerful side effect of the INTERSECT-REMOVAL process (section 3.5) is that regardless of how the designer forms a region, two equal shapes have identical sets of non-intersected rectangles. It suffices to store only the size and relative placement of the subregion's nonintersected rectangles and connections. The complete subregion description library is stored in a tree structure as shown in figure 4-13. Each tree node represents one shape description or *clue*. The node contains all valid answers to the node's clue; each valid answer then has a tree branch

not covered by any segments are made into straight resistor subregions, as illustrated in figure 4·11(d).

Figure 4·11 also illustrates two other points worth mentioning. First, we note that the leftmost straight subregion of figure 4·11 has zero length and thus has no resistance to calculate. It does, however, serve to break the regions to the left and right on an equipotential line—something that should be done whenever possible, for it can only simplify computation. Secondly, while the leftmost non-straight subregion could itself be treated as another straight region, the algorithm does not naturally place it as such. It does however match the last resistor library entry listed in table 4·1. This condition typically happens at MOS transistor source and drain connections.
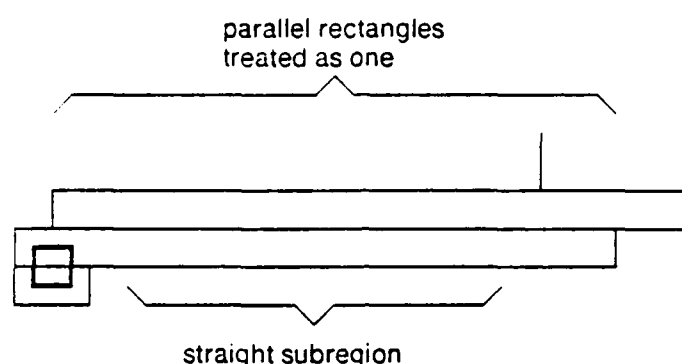
Figure 4·12: Straight regions hidden across two parallel rectangles.

The extractor makes a final pass to isolate straight subregions that are hidden across parallel rectangles like the ones in figure 4·12. Parallel rectangles of this sort are formed only in the horizontal direction as a consequence of the INTERSECT-REMOVE process. When the extractor finds parallel rectangles, it simply treats them as one.

4.5.2 Storing and Locating Library Subregions

After the straight subregions are removed from the whole of the interconnection area, the remaining rectangles are regrouped into isolated islands of connecting rectangles.[9] The islands form all of the other subregions, and we now face the task of recognizing the library subregions.

---

[9] If abutment information is retained, this amounts to a transitive closure search.

74

4.5.1 Locating Straight Subregions

First. the extractor locates straight subregions and removes their area from the whole area. *Internal connections* are added between the straight subregion and the remaining area.

Recall from section 3.5 that the processed. path geometric database contains (1) a record for each rectangle. and (2) a list of all abutments between rectangles. Also recall that the abutments between rectangles are always horizontal. Each rectangle is checked individually for straight subregions following one of two program subroutines. depending on whether the rectangle's height or width is greater. The subroutines' internals are similar: only the use of x and y coordinates are interchanged. We will only consider the procedure for rectangles with greater widths.
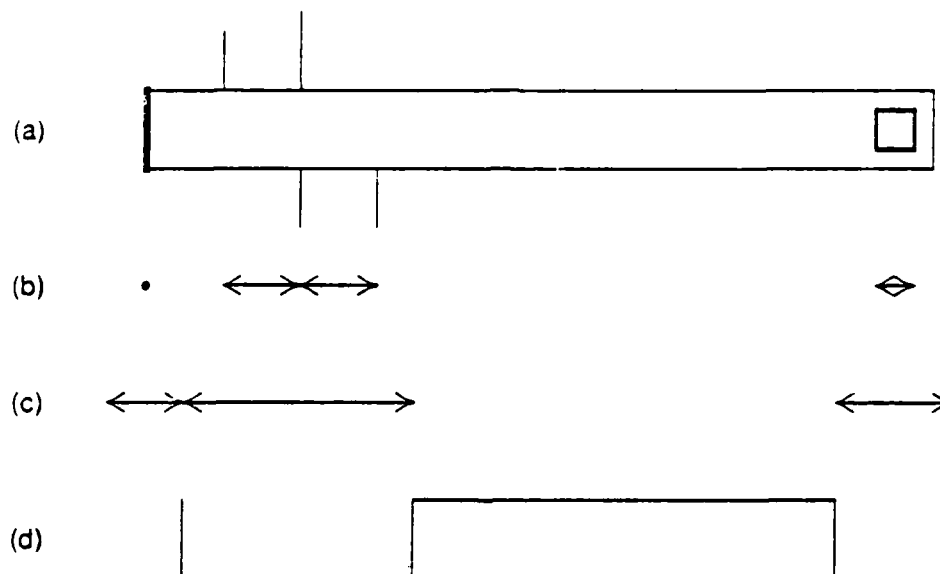
Figure 4-11: Segment operations for isolating straight subregions

The extractor computes the x-coordinates of all abutting rectangles and connections and stores them as *segments*. The extractor expands each segment in both x directions by an amount $\gamma_{cs} \times height$ (see section 4.3.1 for a definition of γ_{cs}) and then combines segments that overlap. The process is illustrated in figure 4-11 for the rectangle shown in (a). The unexpanded segments are shown in (b), and the expanded segments for $\gamma_{cs} = 0.5$ in (c). The expanded segments cover x-coordinate areas that may contain vertical current components. and therefore. rectangle regions

73
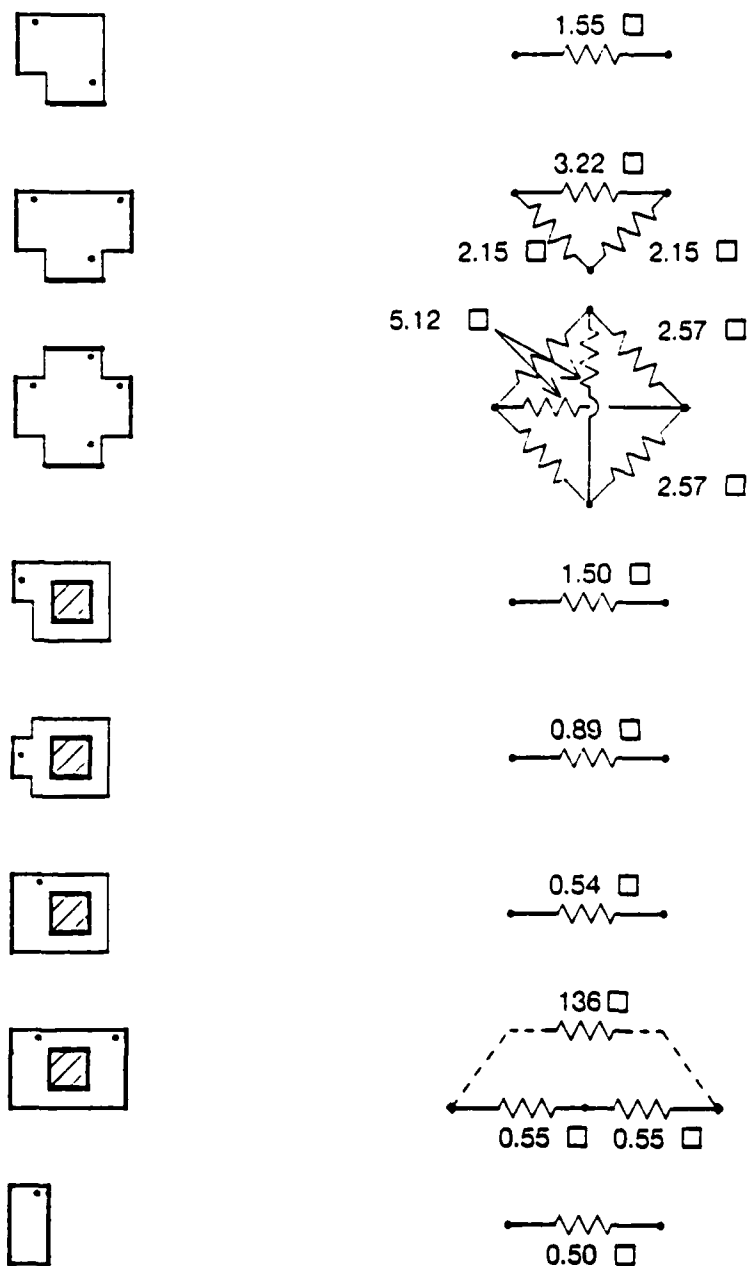
Table 4-1: Sample library shapes and equivalent networks for NMOS process

A rectangle marked by an asterisk (*) has a shape determined
by the current spreading ratio, γ_{cs}.

4.10(b) and (c). The two long straight resistors have the maximum possible current spreading at one end. A conducting strip is placed at the edge of the current spreading region on one resistor, while nothing is placed on the other resistor. By measuring the difference in overall resistance one can estimate the maximum error introduced at the straight resistor's end regions. For γ_{cs} = 1, the error is less than 0.1%; for γ_{cs} = 0.5, the error is approximately 2%. EXCL uses a γ_{cs} of 0.5 throughout.

4.4 Stored Calculations of Commonly Occurring Shapes

The final technique for finding resistance requires no calculations of fields, dimensions, etc. during extraction, for this is done in advance. The equivalent resistor subnetworks of certain shapes that occur frequently in IC's are stored in the *resistance library* of the extractor program. When the the extractor recognizes a shape to any scale, it merely recalls the library resistor subnetwork, and connects it to the main network. The library requires a shape to be easily recognized, and to have connection points to other subregions only on equipotential lines. The second restriction is more prohibitive, and essentially limits the internal connections to straight resistor subregions.

Table 4-1 shows some resistor library shapes for the NMOS process. The first three are formed by intersections of straight regions and are obvious members of this list. The others are common contact cut configurations; the list of these is certainly not exhaustive. The shape "recognizer" is programmed to recognize these shapes at any scale, *x* or *y* reflection, or orthogonal reflection. The network is the same, regardless. The next section examines the methods of storing and recognizing library shapes.

Computing equivalent networks of library shapes certainly does not need efficiency—in fact, the desire here is accuracy. Numerical, finite element methods (section 4.2) with extra high accuracy were used to generate some network values found in the table. Others were calculated with great precision by Hall [19] using conformal transformation methods.

4.5 Subdividing Geometric Regions

Thus far in this chapter, we have examined three different ways of calculating resistance from a geometric region. Due to the great simplicity of the latter two techniques—calculation of straight resistors and stored calculations from know shapes—the extractor uses these whenever possible. Only when the region is so irregular that neither of the simple techniques work does the extractor rely on the backup finite element techniques. In this section we consider how the extractor recognizes subregions to be computed by one of the three techniques.

71

resistor as a region with uniform width and with current flow everywhere parallel to the length. This reduces the field problem to a one-dimensional, linear problem. One can easily show that the field equations of section 4.1 reduce to

$$R_d = \rho_{sn} \frac{L}{W} \qquad (4.19)$$

for a straight resistor region with constant sheet resistance ρ_{sn}, a length L, and a width W.

Error analysis for the resistance calculation of equation (4.19) is not needed, for the equation is exact. However, we must consider the errors introduced by the bending fields at either end of the resistor.

4.3.1 Current-Spreading Region

The uniform current flow in the straight resistor is disturbed by the jog, bend, or curve, at the end of the region. A *current-spreading region* of length $L = \gamma_{cs} \cdot W$ is subtracted from both ends (figure 4-10(a)), and we assume that all current flow distortion is limited to this area.
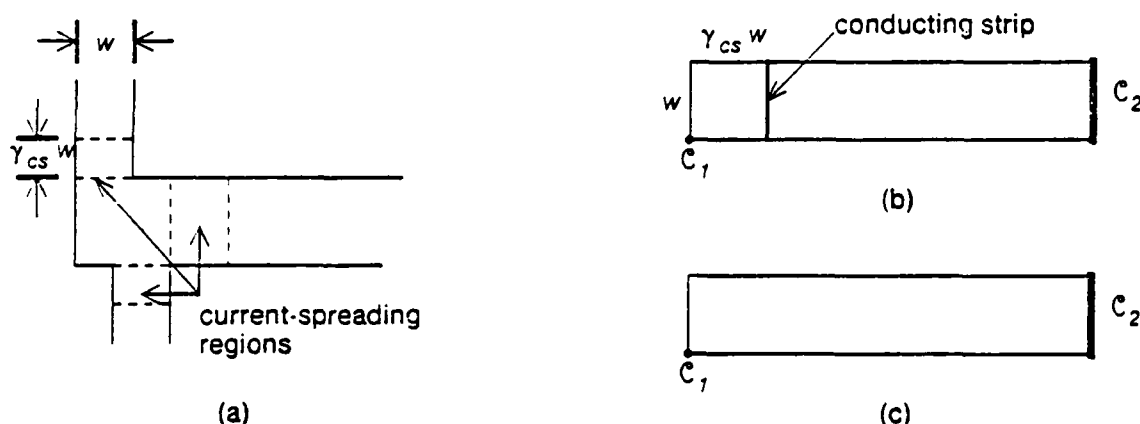
Figure 4-10: Current-spreading regions and maximum error estimates

(a) shows the current spreading regions at the junction of three straight subregions. The shapes in (b) and (c) are used to measure the maximum error. (b) estimates the calculated resistance, while (c) estimates the real resistance.

The worst-case accuracy of this assumption is measured by the experiment shown in figure

70
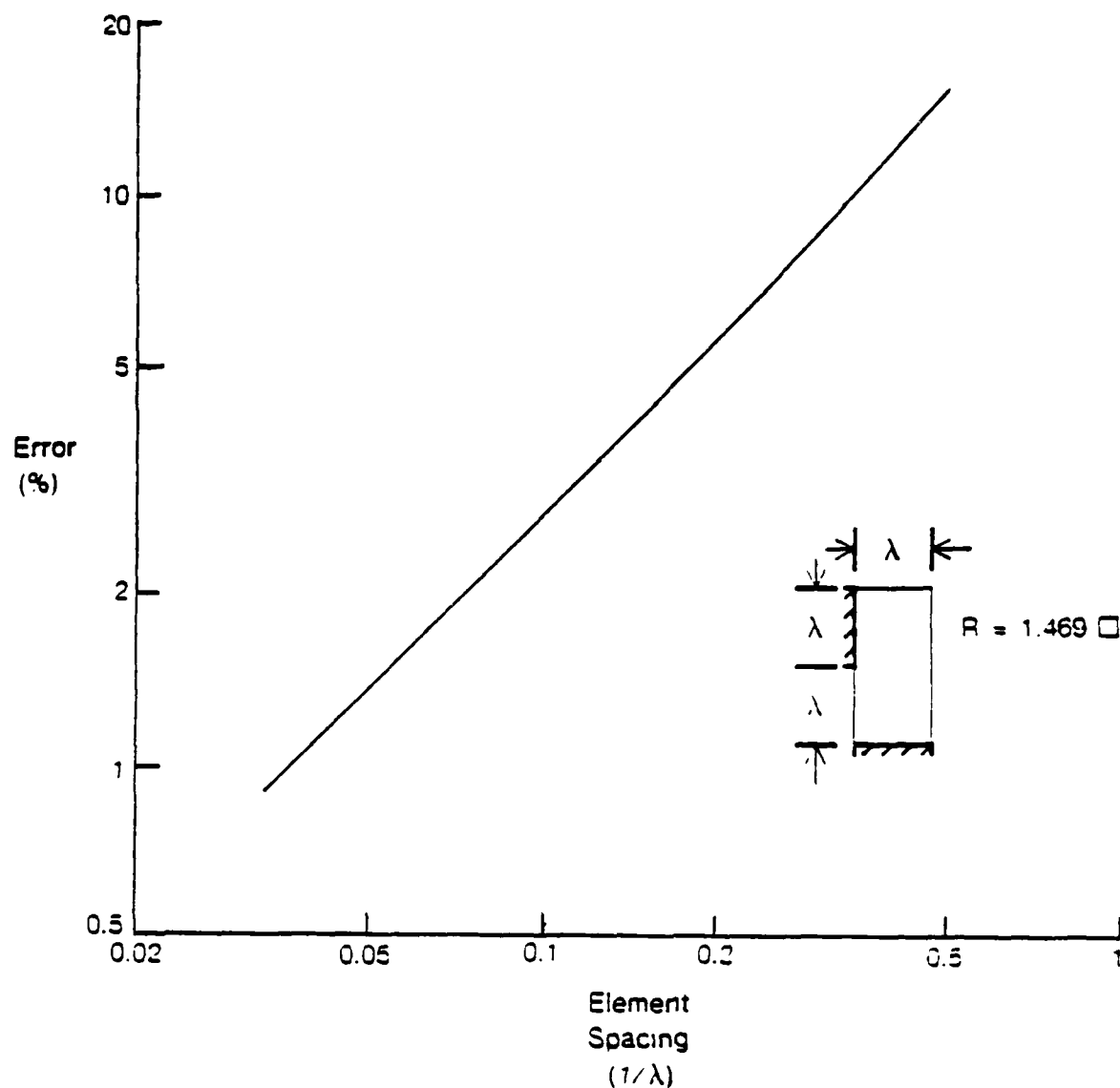
Figure 4-9: Discretization error vs. finite element separation

4.3 Resistance Calculation of Straight-Subregions

As mentioned at the start of this chapter, the extractor uses three calculation techniques to find resistance. While one can apply the finite element technique to any shaped resistive region, its computation requirements are enormous for large regions. Hence, whenever possible the extractor uses the other two techniques that have substantially faster calculation methods.

Both of the alternate techniques solve the resistance problem for shapes that commonly appear as interconnections in IC's. The first of these is the most prevalent shape—the straight wire, also denoted in this chapter as the "straight resistor" or "straight subsection". We define the straight
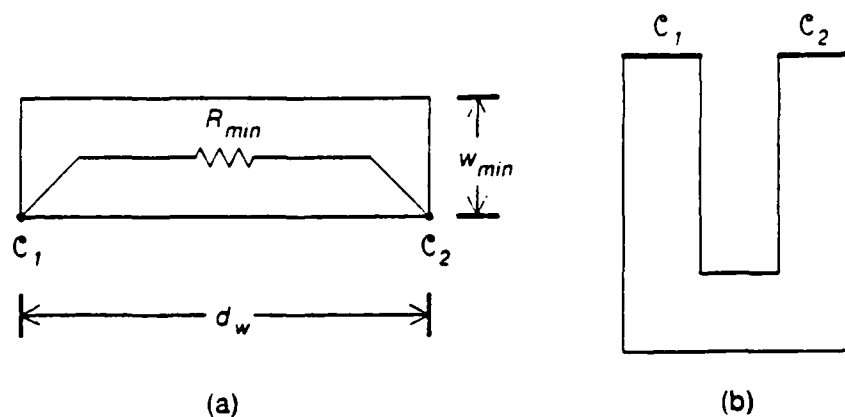
Figure 4-8: Conditions for combining connection regions

(a) condition for worst case resistance between connections.
(b) illustration showing the need for condition 2.

$$d_w = (R_{min} - 3.4) \, w_{min}$$

where R_{min} is in $\square$'s and w_{min} is the minimum mask geometry defined from the technology's design rules and from the maximum mask offset given to the extractor.

4.2.6 Accuracy of the Finite Element Method

As the spacing between finite elements approaches zero (and the element count approaches infinity), the discretization error diminishes. Little theoretical analysis has been made to determine the error for a given element spacing—we must rely on experimental data. Figure 4-9 plots the discretization error versus the finite element spacing for a constant shape. Since the discretization error is greatest at corners, the experimental shape is a corner of known resistance value [19]. Results show that the error is proportional to the finite element spacing. EXCL allows the user to set the element spacing to achieve his own accuracy requirements.

modelled as a "completely connected" resistor network. as shown in figure 4-7 for a region with four principle boundaries. A test voltage connects to each conductor. The superposition principle aids in the network analysis. To find the values of all resistors connected to a certain conductor. say C_1, we set V_{T1} to 1 volt and set all other test voltages to zero. After computing the voltage field on $\mathcal{R}$. we find the current flowing from C_1 to any of the other conductors. C_k by defining the surface P of equation (4.6) as the boundary between $\mathcal{R}$ and C_k. The computation of R_{1k} follows directly from the current, I_{1k}.

(a)   (b)
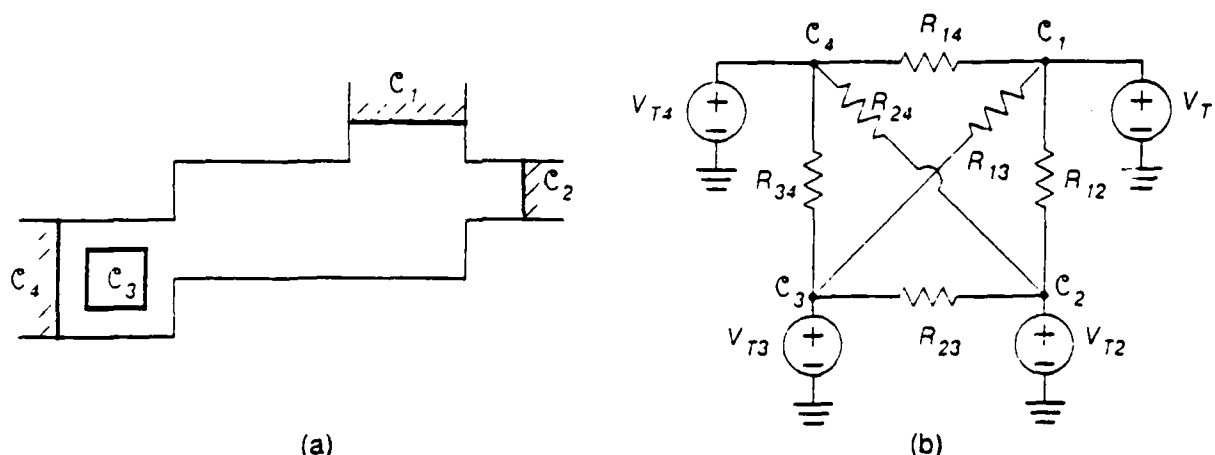
Figure 4-7: Completely connected resistor network

(a) resistive region with four connections,   (b) equivalent resistor network

To find all resistor values for a network with m conductors requires $n - 1$ calculations of v and $\frac{(n)(n-1)}{2}$ calculations of I_d. For a region with large m, this means a lot of computation! Any chance to reduce m is welcomed. Such is provided if the extractor is given a minimum resistance, R_{min}. Some of the connections will eventually be combined by shorted resistors. While recognizing shorted regions before extraction is not always possible, often it is possible by examining the spacing between two connections. If the following conditions are true, we know that R_d is less than R_{min}:

1. The distance between conductors of R_d is less than the maximum distance for worst-case conduction, d_w.

2. A straight line segment can be drawn between the two conductors that is always in $\mathcal{R}$.

The worst-case conduction distance is derived from the situation shown in figure 4-8(a). For $R_{min} >$ $4\square$'s,

67
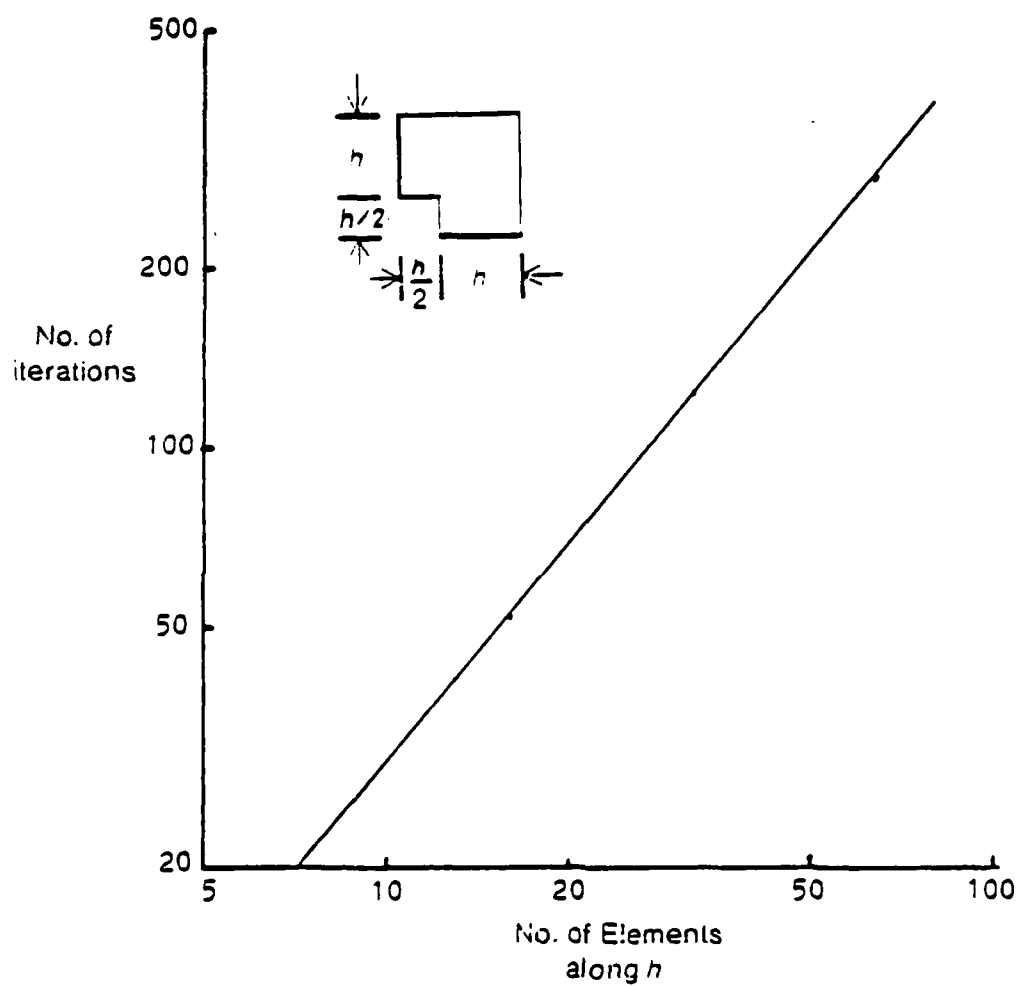
Figure 4-6: Number of iterations vs. one-dimensional finite element count.

66

5.1 Inter-Nodal Capacitance

The inter-nodal capacitance extractor, EX-COUPLING, is called by the connectivity extractor and calculates capacitance values between one *target conductor* and a set of other *source conductors*. All source conductors lie in the same conducting layer, but not necessarily the same layer as the target. For instance, we might be computing capacitances between a "polysilicon" target conductor and all "metal" conductors. The "metal" conductors in this example comprise the source conductors. The connectivity extractor passes process dependent capacitance information as a set of constants and tabular functions.

The connectivity extractor composes a completely connected capacitance network from the values returned by the capacitance extractor. For a three conductor system, the complete capacitance network has the form shown in figure 5-2. The C_{ii} capacitors are ground capacitors, and are included here for completeness. All conductors take turns being the target conductor. For any given target, the extractor is capable of finding values for all capacitors connected to it, but some values may already be known since $C_{ij} = C_{ji}$. To save the extractor from recalculating an inter-nodal capacitance, a source conductor is tagged if the capacitance between it and the target is known.
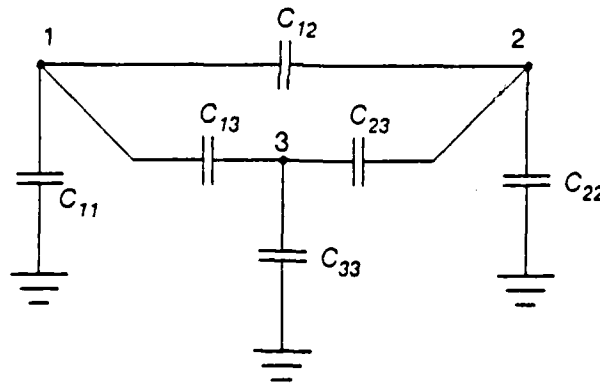
Figure 5-2: Complete capacitance network for a three conductor system

The inter-nodal capacitance "domain" for any conducto, is limited to a small region around the conductor. It is essential to use this observation to avoid a massive capacitance network that grows with the square of conductor count. The connectivity extractor locates source conductors by looking through a window that is made slightly larger than the geometric region of the target conductor.

Windowing coupling capacitance regions keeps the capacitance network problem to a linear growth rate. The exact sizing of the window will be examined later.

The basic philosophy of dividing a region into subregions with different classes of solution techniques carries over from the resistance problem to the capacitance problem. Where the field conditions are regular, the extractor applies simple calculation techniques; where the field conditions are irregular, the extractor must rely on general techniques. First, this chapter covers methods of subdividing the capacitive region, then it covers the field theory used in inter-nodal capacitance extraction, and finally, it covers the general and special capacitance extraction techniques.

5.1.1 Subdivision of Capacitance Problem

As with resistance extraction, the inter-nodal capacitance extraction problem benefits greatly from dividing the problem's region into smaller subregions. When possible, the extractor forms a subregion in an area where either (1) the field conditions are uniform and capacitance is quickly solved with a closed form equation, or (2) the capacitance is precomputed. Like the resistive subregions, the capacitance subregions must have two properties. First, the subregion must be easily located by inspection of the conductor geometries. Secondly, the borders between subregions must fall on constant flux lines (perpendicular to $\vec{E}$).

The capacitance extractor uses two classes of simple capacitance calculation. One is applied between overlapping regions of different layered conductors, the other between parallel spans of any two conductors. First, the extractor locates overlapping areas and removes them from the source conductors' regions. The border between the removed and non-removed area forms an *artificial edge*; all other borders at the conductors' mask edges form *natural edges*.

Next, the capacitance extractor locates the parallel regions and library regions with the same procedure used by the resistance extractor. However, the capacitance extractor divides the oxide areas between conductors, as shown in figure 5-3(b). To achieve the correct division, segments are defined and expanded only at the oxide rectangle abuttments and at the corners of conductor rectangles. In the figure, corners are marked with a dot. Refer to section 4.5 for a discussion segment operations for subdivision. The irregular regions include *flux-spreading* areas near parallel subregions. Flux-spreading regions are directly analogous to "current-spreading" regions of the resistance extractor, since they define an area where flux-line bending occurs.

81

Window

source conductor

target conductor

overlapping conductors

(a)

single-side | irreg.

source conductor

irreg. parallel | irreg. parallel | irreg.

target conductor

irreg. | library | single-side
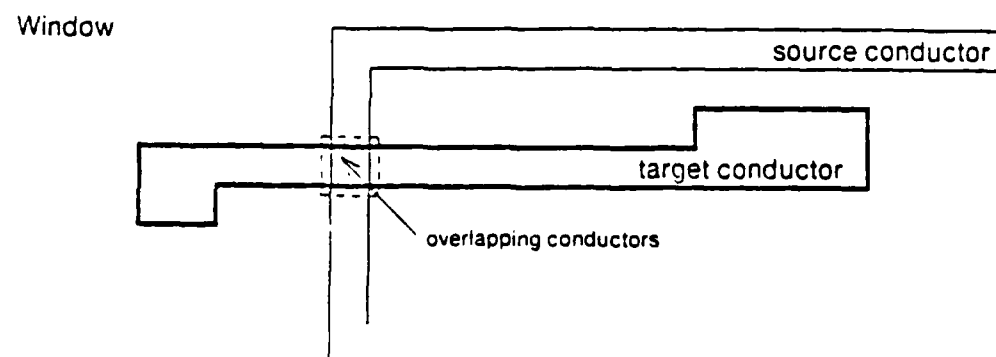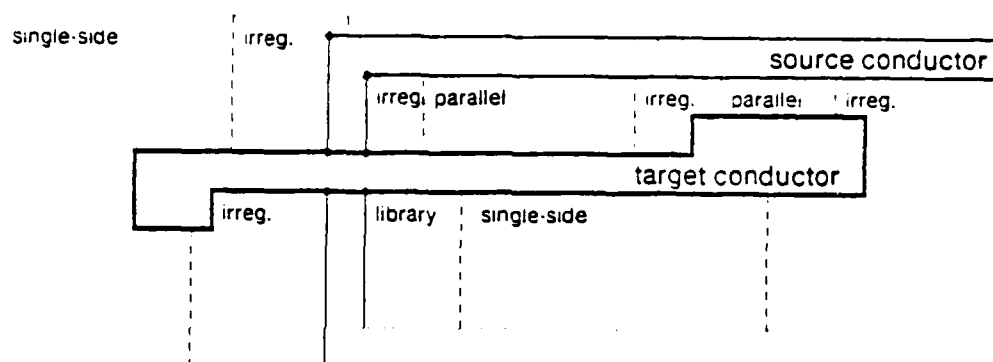
(b)

Figure 5-3: Subdivision of capacitance region.

(a) removal of overlapping area, (b) division into parallel, library, and irregular subregions.

5.1.2 Field Theory of Capacitance Calculation

The physical region of the inter-nodal capacitance problem is divided into three areas. The IC interconnection areas or *conductor* areas form the capacitor "terminals" or "plates" and are denoted by $\mathcal{I}$. The *dielectric* (or *insulating*) areas denoted by $\mathcal{D}$ comprise the oxide between conductors. Lastly, the silicon substrate forms the *groundplane* area, $\mathcal{G}$. The $\mathcal{I}$ and $\mathcal{D}$ regions are

the $\mathcal{R}$ and $\mathcal{C}$ regions, respectively, in the resistance problem. To avoid confusion with the different boundary conditions of the two problems, the names have been changed for this chapter.

To find the inter-nodal capacitance between two conductors, we force a known voltage difference, V_d, between the conductors. By determining the charge, Q, induced on either conductor, we can compute the coupling capacitance, C_c, by

$$C_c = \frac{Q}{V_d}.$$ (5.1)

The electrostatic field equations relating charge and voltage are directly analogous to the equations for electric conduction presented in section 4.1. In making the analogy, the electric potential field, V, remains the same, but current density, $\vec{J}$, is replaced by electric flux density, $\vec{D}$. The relationship between the two fields now depends on the material's permittivity, ε, as

$$\vec{D} = \varepsilon \vec{E} = -\varepsilon \nabla V.$$

The potential in an electrostatic field follows Poisson's Equation,

$$\nabla^2 V = -\frac{\rho}{\varepsilon}.$$

In dielectric regions, however, the charge density, ρ, is zero, and Poisson's Equation reduces to Laplace's Equation,

$$\nabla^2 V = 0.$$

This might suggest solving the general capacitance problem with the same techniques developed for resistance, only viewing the electric flux density flowing through the dielectric medium, $\mathcal{D}$. With this technique, a known potential is placed on all conductors, and the potential field is calculated in the insulator. Then, the total charge on each conductor is found by integrating over its outer surface, P:

$$Q_d = \oint_P \vec{D} \cdot \vec{n} \, dP,$$

where $\vec{n}$ is a unit vector pointing out of P.

While the technique of solving Laplace's Equation is useful for careful capacitance calculations of some special cases, it is not practical for the general case. In the electric conduction case, the field regions have well defined boundaries, but in the electrostatic case, the insulating medium extends unbounded in all directions. An alternate, "integration" approach has been adopted by several existing coupling capacitance extractors [20, 21].

The theoretical basis follows from Green's Theorem which shows that for any point in space, a, its potential is given by

$$V(a) = \int_{\mathcal{J}} G(a; b) \, \rho(b) \, db. \tag{5.2}$$

The integration over $\mathcal{J}$ must include all charge-carrying points, b, as shown in figure 5-4. $G(a; b)$ is the appropriate Green's function for the insulating medium between points a and b. It describes the voltage induced at point a by a unit charge at point b. In free space

$$G(a; b) = \frac{1}{4\pi\varepsilon_0 r},$$

where r is the distance between a and b. In an IC the Green's function is more complex due to the different, finite-thickness oxide layers and the substrate. The substrate contributes charge in the capacitance problem, but rather than extending the integration of equation (5.2) to cover $\mathcal{G}$ as well as $\mathcal{J}$, the Green's function is typically adjusted, instead.
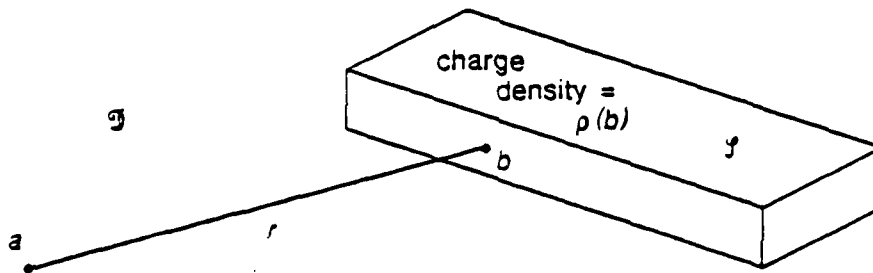
Figure 5-4: Green's Theorem for finding V at point p

5.1.3 Finite Element Techniques

In the existing extractors mentioned earlier, the conductor's top surface is divided into either a regular square grid [22, 21] or a special rectangular grid [20]. Subdividing the conductors in this manner yields good results when the conductor's thickness is significantly less than the spacing between conductors. However, when the conductor's thickness approaches the inter-conductor spacing, the capacitance is underestimated, because, much of the additional coupling between the two conductor edges is not represented.

In section 2.4.2 it was demonstrated that coupling capacitance shows important effects only when coupling capacitance exceeds a certain fraction of the total capacitance, or when

84

$$C_c > \gamma(C_c + C_g). \tag{5.3}$$

We now consider a typical case where the conductor width equals the conductor spacing (or $W = S$) and the conductor thickness equals the conductor height (or $T = H$) as shown in the cross-sectional view in figure 5-5. Results from Dang [23] show that the condition in equation (5.3) is met with $\gamma = 0.1$ only when the ratio of conductor thickness to conductor spacing exceeds a value of 0.25 (or $T/S > 0.25$). As the T/S ratio increases above 1.0, the coupling capacitance exceeds the ground capacitance. While the tendency is to increase T/S to reduce the conductor's resistivity, this makes the coupling capacitance between conductors more unattractive.
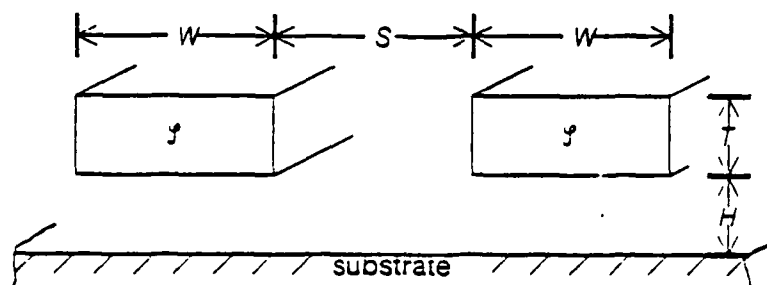
Figure 5-5: Cross-sectional view of conductor geometries.

As T/S increases, the amount of coupling capacitance due to conductor edge surfaces increases. The approach taken in EXCL for extracting inter-nodal capacitance assumes most of the coupling is between conductor edges or between the top and bottom surfaces very near the edge. In EXCL, the conductor edges are subdivided into N finite elements, $e_1, e_2, ..., e_N$, and N nodes, $a_1, a_2, ...,$ a_N, as shown in figure 5-6. Associated with each finite element is a region of the top and bottom conductor surface directly behind the edge. At artificial edges formed at conductor breaks near overlapping conductors only the top and bottom surface is considered in the finite element. Over each finite element EXCL assumes a charge distribution, $\rho(b) = q \cdot f(b)$, where b covers the bottom, edge, and top of each element. q is the total elemental charge, and f is a *shape function*. Some sample shape functions are illustrated in figure 5-7; each one describes how a unit charge spreads over the finite element.

Equation (5.2) may now be rewritten into a summation over each of the N finite elements:

$$V(a) = \sum_{i=1}^{N} \int_{\text{element } i} G(a; b)\, \rho(b)\, db = \sum_{i=1}^{N} q_i \int_{\text{element } i} G(a; b)\, f(b)\, db. \tag{5.4}$$

85
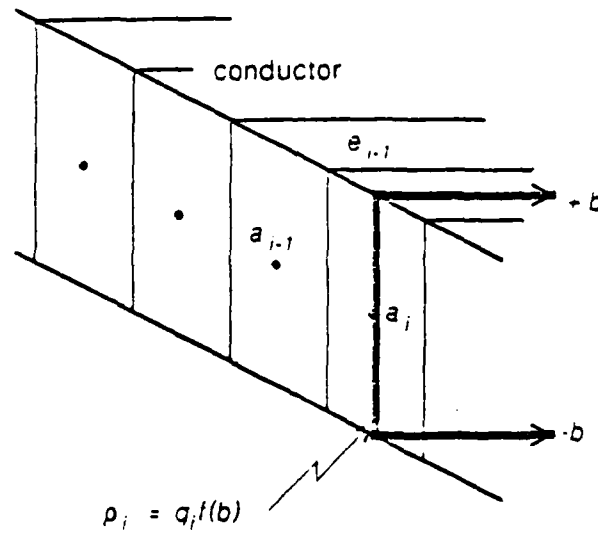
Figure 5-6: Finite element for inter-nodal capacitance extraction.

We define the *weighted-Green's* function,

$$H(a_j; e_i) \triangleq \int_{\text{element } i} G(a_j; b) \, f(b) \, db, \tag{5.5}$$

which describes the voltage induced at a point a_j by a unit charge distributed over finite element e_i. By combining equations (5.4) and (5.5) we get

$$V(a_j) = \sum_{i=1}^{N} q_i \, H(a_j; e_i). \tag{5.6}$$

We now write a potential equation (like equation (5.6)) for each node point, $a_1, \ldots, a_N$. Expressing the system of N linear equations in matrix form yields

$$v = H \cdot q,$$

where H is a matrix with entry h_{ij} equal to the weighted-Green's function value $H(a_j; e_i)$.

In the capacitance problem, the electric potentials, v, are known and the charges, q, are unknown. Hence, we actually solve for

$$q = H^{-1} v$$

using any of the standard matrix techniques. Once q is computed. the elemental charges on any conductor are summed to give total conductor charge. From this. the inter-nodal capacitance is calculated.

To summarize the finite element analysis of arbitrary shapes. the following list itemizes the steps made by EX-COUPLING.

1. The conductor regions. f_1 through f_K. are divided into N finite elements. N is the total number of all finite elements. or

$$N = \Lambda(1) + \Lambda(2) + ... + \Lambda(K)$$

if $\Lambda(k)$ is the number of elements on any conductor k.

2. The $N \times N$ matrix. H. is filled with values of weighted-Green's function. Element h_{ij} of H is the appropriate value relating the charge on element j and voltage on element i. The values of weighted-Green's function are precomputed as discussed in the next section.

3. The $N \times 1$ voltage vector. v. is filled with the conductor voltages assigned to each finite element. Generally, one conductor—say the target conductor, f_1—is assigned a non-zero voltage V_g, while all others are assigned zero. thereby enabling the computation of capacitances between f_1 and all other conductors. As with the resistive analysis. the complete inter-nodal capacitance matrix between K conductors can be computed with $K - 1$ settings of conductor voltage (see section 4.2.5). but only one setting is needed for each invocation of EX-COUPLING.

4. The charge vector. q, is computed by solving the system of equations $v = G \cdot q$. ExCL solves for q by Gaussian elimination with partial pivoting.

5. The total charge on a conductor, f_x, is found by summing all values of q that correspond to finite element charges on f_x.

6. Lastly. the capacitance between f_1 and f_k is calculated from equation (5.1).

5.1.3.1 Determination of Weighted-Green's Functions

As mentioned earlier. the Green's function between two points in an isotropic medium separated by r is $1/4\pi \varepsilon r$. but in IC's. Green's function is complicated by the oxide layers and the substrate. Efforts have been made by Silvester and Patel [24, 21, 22] to characterize the Green's functions for a thin conductor on a dielectric sheet by methods of partial images and Fourier integrals. These methods have been severely limited in modelling groundplanes. multi-leveled conductors, and

conductors with non-zero thickness. Direct measurement of Green's function is not possible, for no IC structure generates a charge at a single point or finite element.

A method presented here for determining a weighted-Green's function uses a series of two-step computer field simulations of the known IC structure. For a given IC process, the functions are computed once: thereafter, they are stored in lookup tables for easy reference. If necessary, the functions can be tweaked to fit measured data. A different weighted-Green's function is used for each pairing of conductor layer. i.e., poly-poly, poly-metal, ... One function is determined with each two-step simulation. The first step finds an approximation to $f(b)$, while the second finds an approximation to $H(a; e_i)$.

The shape function derivation begins with a simulation of the voltage field around two parallel conductors with a non-zero voltage difference applied to one conductor, as illustrated in figure 5-8(a). The charge induced on the other conductor is measured. The charge distribution is then uniformly scaled to fit a unit charge onto a finite element. The field simulator uses the Laplace Equation methods discussed briefly in section 5.1.2 and presented more thoroughly by Dierking [25] and Dang [23]. The simulator also includes enhancements for infinite boundaries as developed by Dierking and for boundaries between different dielectric materials. The shape function depends on the conductor spacing in addition to the layer pair, so the simulation steps are repeated at several spacing intervals. The shape functions of figure 5-7 were computed by the field simulator for the conductor configurations illustrated.

The second-step of the simulation computes the voltage field around a unit charge distributed in the shape approximated by the first simulation. This simulation employs spherical coordinates in the direction away from the conductor, since the effects from a single finite element away from the conductor are approximately spherical. Appendix C presents techniques developed for spherical simulation. Figure 5-8(b) shows the $\theta = 0$ plane of a unit charge simulation for the shape function derived in figure 5-8(a). With this simulation we can compute the weighted-Green's function for the voltage point, a, with a height h above the substrate, and a unit charge distribution, $f(b)$ on e. The values of voltage. $V(r)$, along line AB are exactly the values of weighted-Green's function, $H(a; e)$, for a horizontal separation r between a and e.

For any two-conductor system, the extractor uses not just one weighted-Green's function, but a series of six functions. Each function represents a different vertical height combination between

88
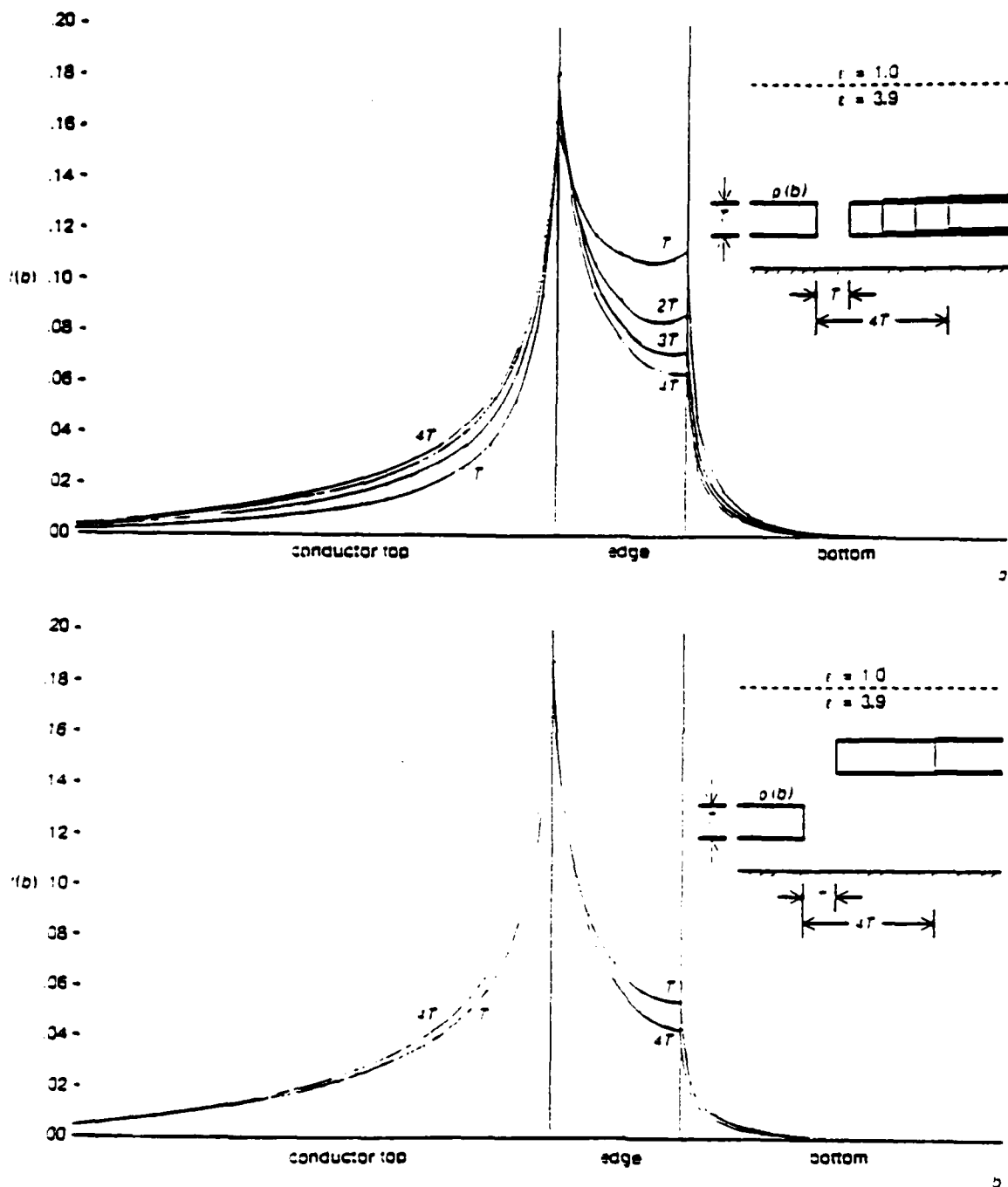
Figure 3-7: Shape functions for conductor shapes shown in inset
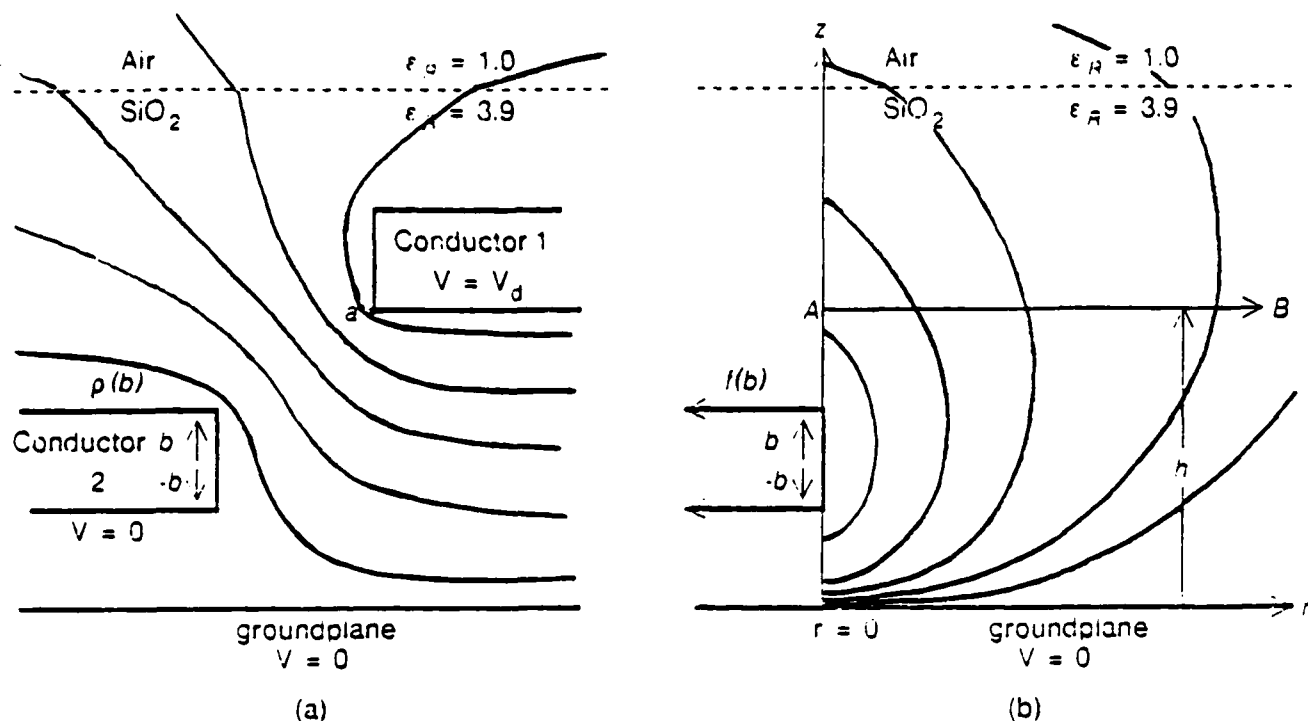
Figure 5-8: Field simulations for determining weighted-Green's functions.

(a) Shape function simulation for finding $f(b)$ on conductor 1.
(b) Voltage field simulation of a unit charge distributed on conductor 1.

charge element and voltage point. They also distinguish between finite elements formed on natural edges and artificial edges. The six functions are listed in table 5-1 along with the field simulation configuration from which they are derived. Figure 5-9 shows a plot of sample weighted-Green's functions for a constant shape function. For the case where the source and target conductor are on the same conducting layer, the extractor needs only two functions, $H_{11} = H_{22}$, and $H_{12} = H_{21}$. Artificial edges never form when the source and target conductors are on the same layer, and thus H_{13} and H_{23} are unneeded.

Each weighted-Green's function value depends on three factors. One factor—the conductor pair combination—is handled with the techniques presented in the last paragraph. The other two factors are the shape function, f, and the horizontal distance, r, between the voltage point and finite element. Both enter into the selection of the value, $H(f, r)$. The extractc actually decides on a shape function with the distance to the "dominant" or nearest point. Each function, $H(f(dominant\ pt.), r)$, is stored in a two-dimensional table with linear interpolation used for lookup of intermediate values.

90
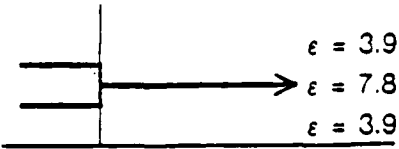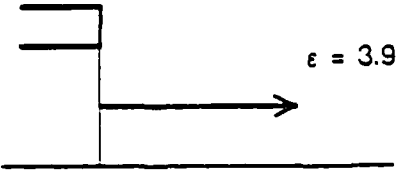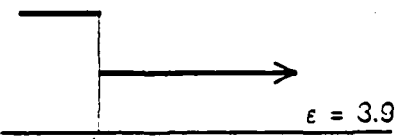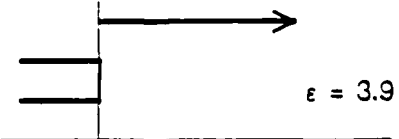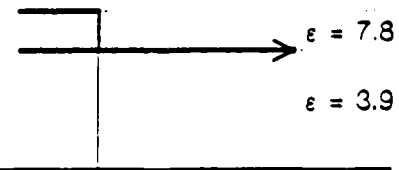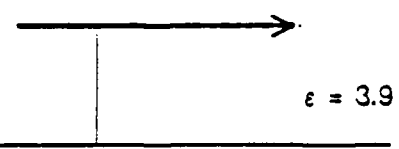
Function	a	e	simulation
H_{11}	target conductor	target edge	$\epsilon = 3.9$ $\epsilon = 7.8$ $\epsilon = 3.9$
H_{12}	target conductor	source natural edge	$\epsilon = 3.9$
H_{13}	target conductor	source artificial edge	$\epsilon = 3.9$
H_{21}	source conductor	target edge	$\epsilon = 3.9$
H_{22}	source conductor	source natural edge	$\epsilon = 7.8$ $\epsilon = 3.9$
H_{23}	source conductor	source artificial edge	$\epsilon = 3.9$

Table 5·1: Series of weighted-Green's functions for two conductors on different layers.

All examples in this chapter have, thus far, been for conducting layers above the IC substrate. The methods also serve for diffusion conductors imbedded in the IC substrate. Figure 5·10 depicts the simulation conditions for determining diffusion functions. The dielectric region below the diffusion conductor represents an approximation to the space-charge layer of the back-biased diode. Because of their close proximity to the groundplane, diffusion conductors tend to exhibit much less coupling capacitance and more ground capacitance than other conductors.
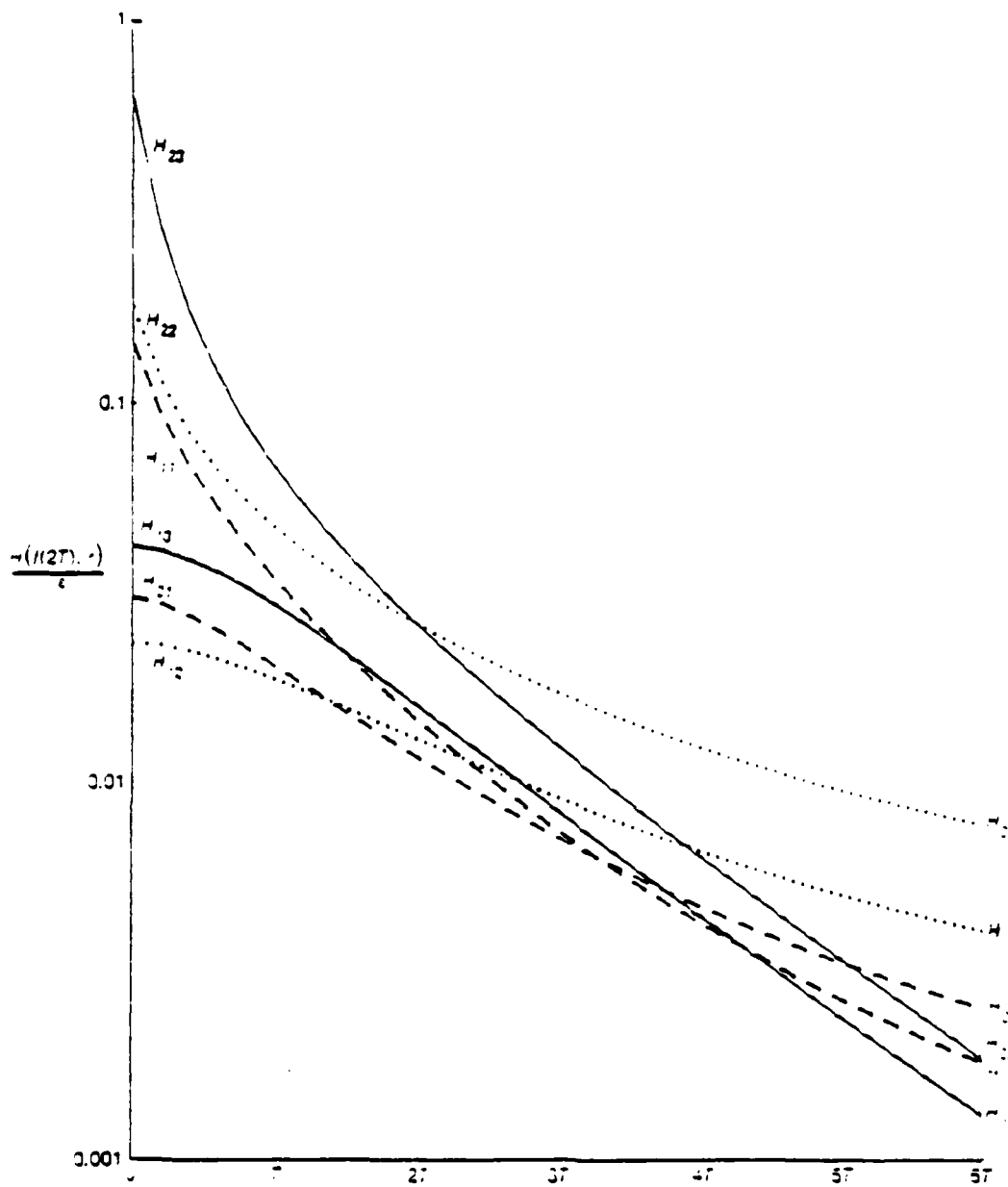
Figure 5-9: Weighted-Green's Functions vs. distance for a constant Shape Function
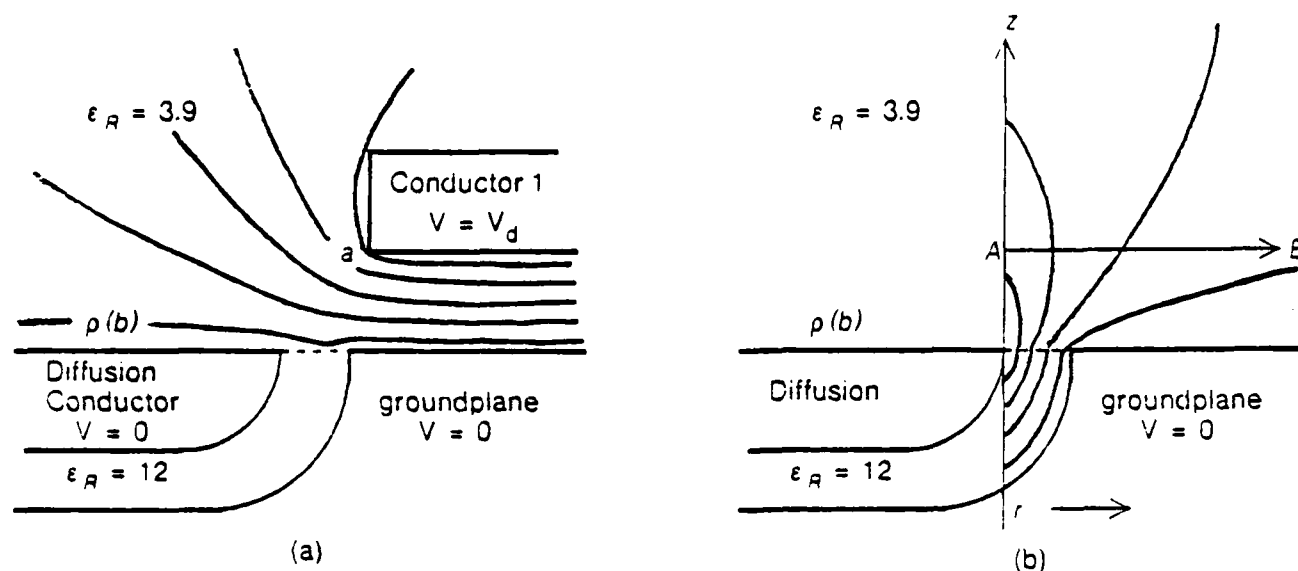
Figure 5-10: Simulation boundaries for diffusion conductor

(a) Shape function simulation, and (b) unit charge simulation.

5.1.3.2 Sources of Error in the General Method

Each of the following four factors introduce error into the general, inter-nodal capacitance method:

1. The shape function calculation assumes that the conductor width behind the primary edge is infinite, which is of course false. One can estimate an upper bound on the error's magnitude by determining what fraction of the shape function lies beyond the actual back edge.

2. The unit charge effects on a finite element do not posses exact spherical symmetry, as was assumed earlier. At large angular deviations from the edge normal, triangulation effects from the top and bottom charge distribution may become noticeable.

3. The selection of shape function from the "dominant" point is accurate only for parallel conductors, but deviates slightly under irregular conditions. A broader-scoping selection of the shape function would improve this problem.

4. Field simulations that determine weighted-Green's functions are based on measured or estimated geometries. Often the simulations assume orthogonal conductor edges, when
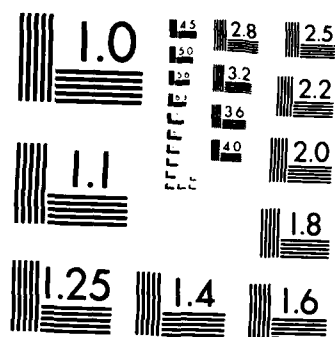
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

in reality they are not. The simulations are accurate only to the degree that the measurements and shapes given to the simulator are accurate.

The first three errors are due to the charge approximation on the top and bottom surfaces. If the amount of charge on these surfaces is small then the errors are likewise small. When the coupling capacitance is greatest. most of the coupling charge is on the conductor edges. which produces less error.

5.1.4 Overlapping Conductors

The capacitance between two overlapping conductors follows the simple equation describing parallel plate capacitance:

$$C = \frac{\epsilon_{oxide} \cdot Area}{Separation}.$$

The values of ϵ_{oxide} and Separation are fixed for a known layer pair. and therefore, we can combine them into a single constant. $K_{overlap}$. An additional "fringe correction". α_{fringe}. is applied to the perimeter of the overlapping areas. giving

$$C_{overlap} = (K_{overlap} \cdot Area) + (\alpha_{fringe} \cdot Perimeter).$$

The fringe correction not only adjusts for field bending near the outer edges of the overlapping area. but it also corrects a problem with the general inter-nodal capacitance method that has not previously been mentioned. The problem occurs at boundaries between irregular subregions and special subregions. The general capacitance method does not account for the parallel plate field. and thus allows its flux lines to spread over into the overlapping field area. The resulting overestimate of capacitance is proportional to the perimeter of the border, and a negative component of α_{fringe} corrects the problem.

The value of α_{fringe} is determined with a field simulation experiment. The Laplace Equation field simulation of figure 5-11(a) finds the true value of capacitance (per unit border length), while the dual simulation of figure 5-11(b) finds the capacitance as the extractor does. In (b), the dotted lines represent the parallel plate flux lines. and the dashed lines represent the "irregular subregion" flux lines. The difference in capacitance between the second and first simulation equals the value of α_{fringe}.

94

Conductor 2   Conductor 2
                              overlap   irregular

Conductor 1   Conductor 1

groundplane   groundplane
    (a)   (b)
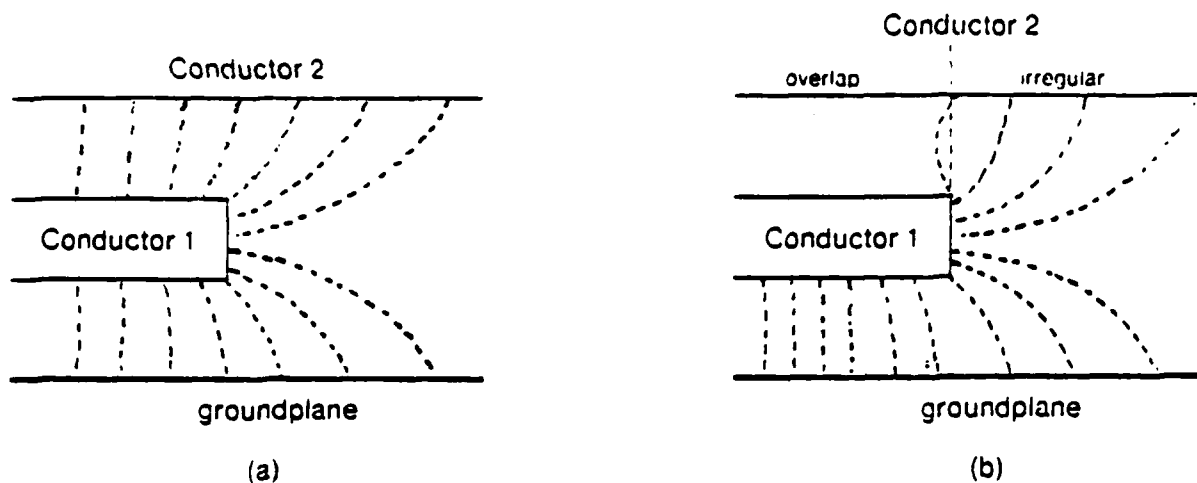
Figure 5-11: Cross-section view of simulations for determining the fringe correction factor

The rectangular coordinate simulations find the capacitance
per unit border length. Simulation (a) finds the true
capacitance value around the border; (b) finds the capacitance
actually computed for the same area.

5.1.5 Parallel Conductors

The capacitance extractor calculates parallel conductor coupling capacitance with:

$$C_{parallel} = \left(K_{parallel}(Spacing) \cdot Length \right) + 2\alpha_{end}(Spacing). \qquad (5.7)$$

$K_{parallel}$ is the *parallel capacitance constant* given in capacitance per unit length. The value of $K_{parallel}$ depends on three factors: the conductor layers, the conductor spacing, and the conductor widths. Like the general case, conductor width is ignored, for we assume that an insignificant amount of charge resides near the far edge. For a given pair of layers, the extractor locates the parallel capacitance constant in a tabular function $K_{parallel}(Spacing)$, where *Spacing* is the conductor-to-conductor distance. The functions are determined either with Laplace Equation computer simulations of the IC geometries, or with direct measurements from test circuits. The example parallel capacitance constants of figure 5-12 were determined with simulations. Typical $K_{parallel}(spacing)$ functions show an inverse linear dependency for very small conductor spacings where the two conductors appear like parallel plates. For larger spacings, capacitance drops off more rapidly.

The "end correction factor", α_{end}, serves the same purpose as the fringe correction factor—it
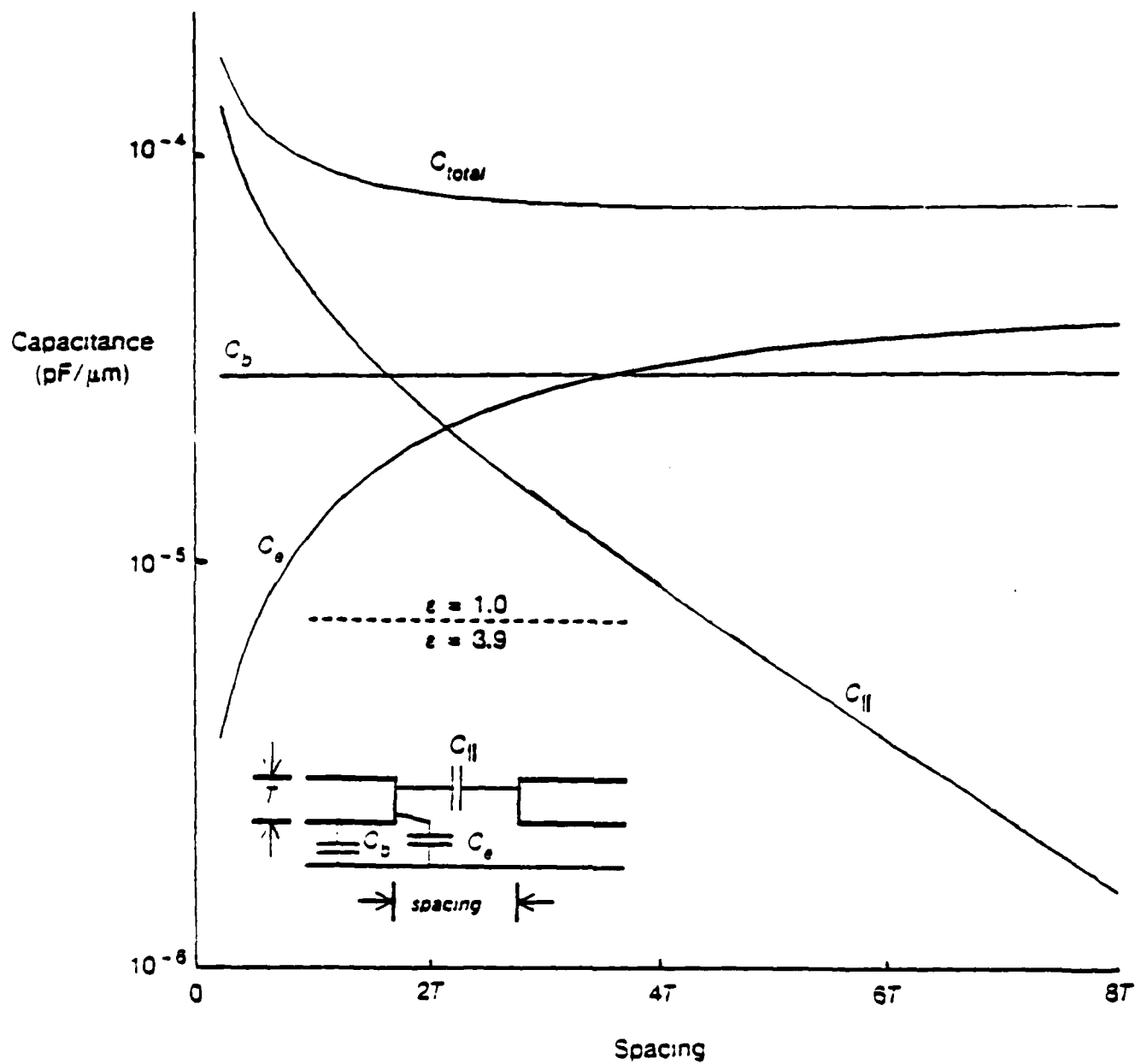
Figure 5-12: Parallel capacitance functions for conductor shapes shown in inset.

compensates for the overestimation of capacitance at subregion borders. This correction factor gives the capacitance overestimation of the general subregion that borders each _end_ of the parallel subregion. The value of α_{end}, which also depends on conductor spacing, is stored in a tabular function, $\alpha_{end}(spacing)$. Figure 5-13 shows the two conditions for computing one value of the function. α_{end} is the difference in capacitance between the two.
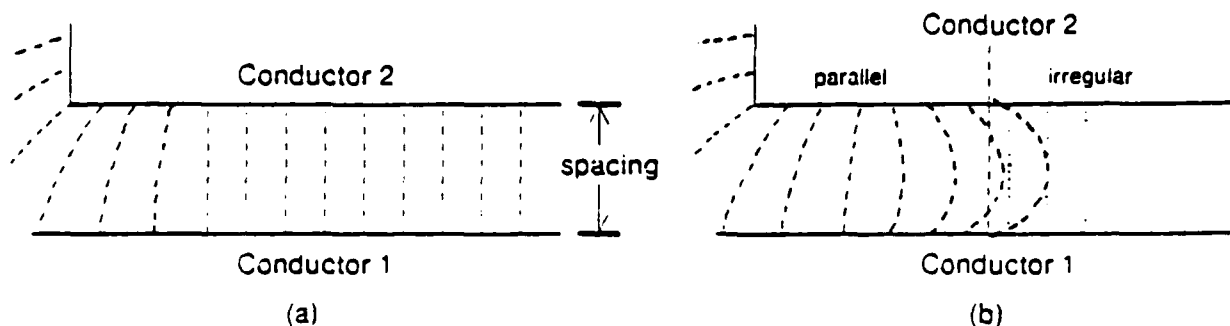
Figure 5-13: Top view of capacitance calculations for determining the end correction factor

We must assume that no field bending occurs at the break between subregions. The true value found in (a) can be computed directly from $K_{parallel}$. The extractor value can be computed by observing the effects of extracting a parallel segment (b) with the general method.

5.1.6 Library Shapes in Inter-nodal Capacitance Extraction

Extracting inter-nodal capacitance with library lookup methods is more limited than extracting resistance with library lookup. Capacitance does not exhibit the same scale independence or layer independence that resistance exhibits. Additionally, useful library shapes are often largely dependent on the window size. Nonetheless, some library shapes have proven useful. Each layer pair combination has its own library with some of the shapes shown in figure 5-14. Some noted entries are useful only for certain pairs.
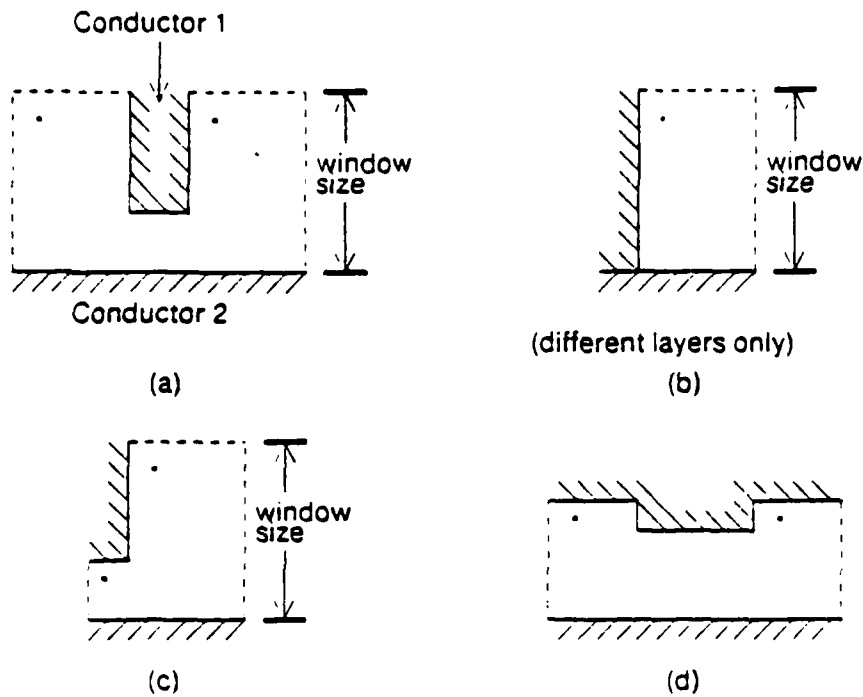
Figure 5-14: Subregions in inter-nodal capacitance library.

Solid lines are conductor edges, dashed lines are borders of
oxide subregions. The size of rectangles marked with an
asterisk (*) is determined by the flux spreading region size. (a)
used for minimum spacing approach of any two layers. (b) used
for different layers only, and usually abutts overlapping regions.
(c) found at 90° bend away from minimum spaced parallel
region. (d) example of metal-to-metal contact cut bulge.

5.1.7 Distribution of Capacitance Among Resistor Nodes

We have seen earlier that the resistance extractor may divide an interconnecting path into
multiple *nodes* connected by a resistor network. The capacitance extractor, however, computes the
inter-nodal capacitance between path as though they were undivided, and then sorts out which
fraction of the capacitance goes to each resistor network node.

Capacitance *distribution* begins by dividing path edges among the resistor network nodes.
Dividing the edges is much easier than dividing path areas. Each path is represented by one or more
node connections as shown by the heavy lines in figure 5-15(a); the connection placement

98

information is passed from the resistance extractor to the capacitance extractor. Recall that many internal connections disappear during minimization of the resistor network (section 4.6). All remaining internal connections extend to the edges on both sides of the path. The points of intersection between the connections and edges are *node-edge points*. Since many of the external path connections (contact cuts, for instance) do not have natural node-edge points, the extractor expands an external connection region until two node-edge points appear. The crosses in figure 5-15(a) mark the node-edge points. After all node-edge points are located, the edges between the points are separated at the half-way point. All parts of the edge are associated with its closest resistor node.

(a)

(b)
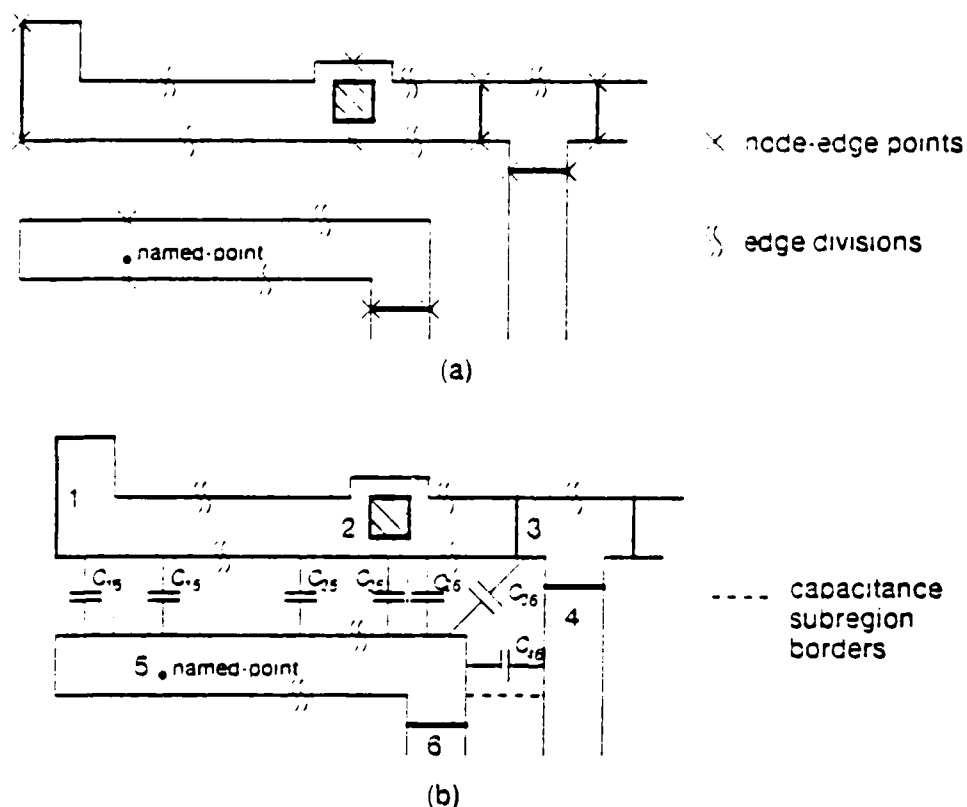
Figure 5-15: Distribution of inter-nodal capacitance among resistor nodes.

(a) Distribution of path edges to each resistor network node.
(b) Distribution of capacitance to nodes.

If a path edge within a capacitance subregion is distributed to more than one node, the extractor must give each node a fraction of the extracted capacitance value. The following guidelines for each

99

type of capacitance subregion control the capacitance distribution. In some cases the guidelines are crude, and give approximate distributions. Figure 5-15(b) shows an example of capacitance distribution.

- In a parallel capacitance subregion the inter-nodal capacitance divides proportionally along its length. The capacitance associated with any node is determined by the length of that node's edge, or

$$C_i = C_{total} \frac{node\ i\ edge\ length}{total\ edge\ length}.$$

If both paths of a parallel subregion have multiple node edges, capacitance distribution is based on the fractional length of any opposing pair of node edges. Figure 5-15(b) shows a sample parallel subregion of this kind.

- The capacitance between overlapping subregions is distributed according to the ratio of each node's edge length to total edge length. This leads to poor division for some irregular overlap shapes, but the important division between long overlapping lines is good.

- Capacitance division of library subregions follows the same rule as overlapping subregions and depends on the ration of the node's edge length to total edge length.

- In irregular capacitance subregions solved with the general method, capacitance division achieves good accuracy. Rather than finding the total path charge before computing capacitance with equation (5.1) the partial charge for each node edge is found and the partial capacitance to each node is computed. When setting up the field problem, however, one must keep in mind to assume that the voltages on different nodes of the same path are identical. We are not interested in find the coupling capacitance between two nodes on the same path.

5.1.8 Window Size Determination

Recall that the coupling capacitance from the target path and any other path is calculated only if the other path lies within the "window" of the target path. To find a path's window, the extractor expands the geometric region of a path by a constant amount, W_x. The value of W_x is calculated such that no path completely outside the window exceeds the condition given in equation (5.3)—including the worst case situation of a source path completely surrounding the target at a distance W_x. If we ignore corner effects, W_x is calculated by examining the coupling capacitance and ground capacitance vs. separation for parallel wires. The separation distance equals W_x at the point where the coupling capacitance is the fraction γ (defined in section 2.4.2) of the total capacitance.

5.2 Ground Capacitance

The capacitance between a conductor and the substrate is very important for circuit extraction. Node extractors for logic simulation typically extract ground capacitance [2]. for special logic simulators can make rough circuit delay approximations with only ground capacitance and transistor size information [8].

The ground capacitance is comprised of two parts. the *edge* and *bottom* capacitance. The sum is most often calculated by

$$C_g = C_{bottom} + C_{edge} = (K_{bottom} \cdot Area) + (K_{edge} \cdot Perimeter). \tag{5.8}$$

K_{bottom} and K_{edge} are constants of capacitance per unit area and capacitance per unit length, respectively. The first product represents the bottom. parallel plate capacitance between the conductor's bottom and the groundplane. Thus. we see that $K_{bottom} = \dfrac{\epsilon_{oxide}}{Conductor\ Height}$.

The second product represents the edge capacitance. and accounts for the capacitance associated with the fringing flux lines between the conductor's edge vicinity and groundplane. The edge capacitance is correct in equation (5.8) only if all flux lines terminate on the groundplane. If a nearby conductor terminates the flux lines. instead. the ground capacitance decreases and coupling capacitance increases. Since the edge-to-substrate capacitance depends on coupling capacitance. it is computed in parallel with inter-nodal capacitance using most of the same algorithms.
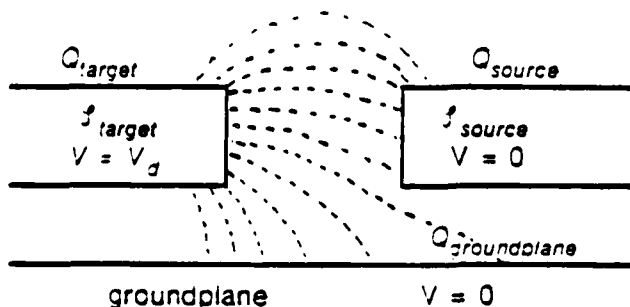
Figure 5-16: Flux lines from edge of target conductor

To observe how the edge capacitance computation operates for the *general* method, consider the situation depicted by the cross-section in figure 5-16. The flux lines originate from the edge of the target conductor, f_{target}, which is set at a positive voltage, V_d. The lines terminate on either the source conductor, f_{source}, or the groundplane; both are set at zero volts. The general inter-nodal extraction method indirectly simulates these flux lines. The coupling capacitance is found by computing the charge on f_{source}. Also available from the general extraction is the edge charge on f_{target}. The magnitude of this charge is

$$|Q_{target}| = \left(\sum_{\text{all sources}} |Q_{source}| \right) + |Q_{groundplane}|,$$

where all quantities represent only the charges induced by flux lines around the edge. We can find the groundplane charge by subtracting the source charge from the target charge, and therefore,

$$C_{edge, general} = \frac{\left(\sum_{\text{all sources}} |Q_{source}| \right) - |Q_{target}|}{V_d}$$

The extractor calculates edge capacitance within parallel and library subregions with the same algorithms that it calculates inter-nodal capacitance for these subregions. In parallel subregions, the edge capacitance is

$$C_{edge, \parallel} = \left(K_{edge}(Spacing) \cdot Length \right) + 2\beta_{end},$$

which we see is very similar to equation (5.7) for parallel coupling capacitance. *Spacing* specifies the same conductor-to-conductor distance, and *Length* specifies the same parallel region length. The correction factor β_{end} compensates for the overestimate of edge capacitance at the subregion borders, and is determined in a like manner to α_{end}. To handle edge capacitance in library subregions, all entries in the capacitance library include an edge capacitance value along with the inter-nodal capacitance value.

Any subregion that was not involved in inter-nodal capacitance computation must be included in edge capacitance calculations. These *single-edged* subregions contain only the target conductor edge (see figure 5-3(b)). Edge capacitance in these regions is $K_{edge}(\infty) \cdot Edge\ Length$, where $K_{edge}(\infty)$ is any value of K_{edge} for very large spacing from other conductors. Finally, it should be pointed out that edge capacitance distribution among path nodes follows the same guidelines for inter-nodal capacitance.

102

5.3 Summary of Capacitance Extraction

This section summarizes the steps involved in all phases of interconnection capacitance extraction, and lists the parameters that are needed. First is a list of all capacitance extraction steps.

The capacitance extractor:

1. finds the edge lengths associated with each node in the target and source paths.
2. creates a capacitance matrix for storing capacitance values for each node. Initially, all values are zero.
3. computes the oxide region between the target path and source paths.
4. locates all capacitive subregions from the oxide region. First the extractor finds overlapping subregions, then parallel subregions, and finally, library, irregular, and single-edged subregions. For each subregion, the extractor:

    a. computes the inter-nodal capacitance with the overlap, parallel, library, or general method. Inter-nodal capacitance is not computed for single-edged subregions.
    b. computes the edge capacitance with the same extraction technique as for inter-nodal capacitance.
    c. distributes inter-nodal and edge capacitance among the nodes in each path.
    d. adds the capacitance components to the capacitance matrix. Edge capacitance is added to the diagonal, ground term for its node.

5. For each node, the extractor:

    a. computes the ground capacitance, and
    b. adds the value to the appropriate diagonal term of the capacitance matrix.

6. Lastly, the capacitance extractor returns the capacitance matrix.

This chapter has involved an extensive discussion of the parameters needed for extraction, and how to derive each. Table 5-2 itemizes all of the parameters as supplied to the inter-nodal capacitance extractor. All derivations of these parameters is done in advance. Some parameters are single-valued, while others are one-dimensional or two-dimensional functions. The capacitance extractor is always invoked for a single, known layer pair combination. The supplied parameters should always match the layer pair, and will change with invocations for other layer pairs.

window size

$H_{11}(f(\text{dominant pt.}), r)$

$H_{12}(f(\text{dominant pt.}), r)$

$H_{13}(f(\text{dominant pt.}), r)$

$H_{21}(f(\text{dominant pt.}), r)$

$H_{22}(f(\text{dominant pt.}), r)$

$H_{23}(f(\text{dominant pt.}), r)$

$K_{overlap}$

α_{fringe}

$K_{parallel}(\text{spacing})$

$\alpha_{end}(\text{spacing})$

inter-nodal capacitance library

K_{bottom}

$K_{edge}(\text{spacing})$

$\beta_{edge}(\text{spacing})$

Table 5-2: Capacitance extractor parameters

CHAPTER SIX

Transistor Size Extraction

Although transistors are the hardest element for simulators to model, extracting their circuit information is easier. For MOS transistors, we typically only need to extract the transistor's length, width, and gate area. Most basic transistor extraction algorithms resemble an algorithm from resistance or capacitance extraction. The notable exception arises when calculating the length and width of an MOS transistor. While the problem is similar to resistivity extraction, it differs in that actual linear dimensions must be found, since transistor effects are non-linear with length.

This chapter describes the algorithms of EX-CONDUCT-DIMENSIONS, that computes the actual dimensions of a channel region between a source node and a drain node—denoted either way as node 1 and node 2. The transistor dimension extractor is supplied with three geometries: the channel region, the edge region (or connection) of node 1, and the edge region of node 2. On rare occasion an MOS transistor may have more than one source or drain. The extraction modeller must resolve this situation with repeated calls of EX-CONDUCT-DIMENSIONS.

6.1 Rectangular Transistors

The vast majority of active regions are designed with rectangular shapes. For some sample IC's examined by EXCL the percentage of rectangular transistors was above 85% in all cases. For these transistors, the extractor wastes no time in determining the length and width from the single channel region.

	normal gauss elimination	band-matrix gauss elimination	band-matrix resolve
triangularization	$\dfrac{2n^3}{3}$	ns^2	ns
backward substitution	n^2	$2ns$	$2ns$
storage	n^2	ns	—
operations for $n = 400$ $s = 20$	4.3×10^7	1.8×10^5	8.0×10^3
storage for $n = 400$ $s = 20$	1.6×10^5	8.0×10^3	—

Table B-1: Summary of approximate operation count and memory needs for gauss elimination

different b's will execute much faster. Furthermore, if the A matrix is unneeded after the first "solving", then no extra memory storage is needed. The row multipliers can be stored in the lower triangular positions for each pivot, occupying the space that would otherwise be zero. Although this feature is not unique to band-matrices, it is included in the band-matrix routines.

B.2 Gauss Elimination with Band-Matrices

The matrix operation most useful for general circuit extraction algorithms is the Gauss elimination solution to a system of linear equations with the form

$$A x = b.$$

The general Gauss elimination algorithm has two steps. *Triangularization* transforms the A matrix into an upper triangular matrix. For each successive "pivot" on the matrix diagonal, the gauss elimination algorithm transforms all entries in the column below the pivot to zero through a series of of row multiplications and additions. For a normal matrix, all rows below the pivot must be transformed, but with a band-matrix, row transformation can stop when the band edge is reached.

During the second step, *backward substitution*, the x values are computed starting with the bottom value. As each new x value is computed, it is substituted into its predeeding row equations. In a band-matrix only, the number of preceding rows where a value must be substituted is limited to half the bandwidth.

Table B-1 summarizes the operation count[10] for general and band-matrix gauss elimination. The table also shows that band-matrix techniques require approximately 0.4% of the general gauss elimination operation count and 5% of the general storage requirements for a 400 × 400 matrix with a bandwidth of 20.

B.3 Repeated Solution with Different Boundary Values

In some cases, we wish to find the solution to several systems of equations with the form

$$A x = b,$$

where A is the same for each solution and only b is different. This situation arises, for instance, when extracting a resistive region with more than two boundaries. We saw in section 4.2.5 that the voltage field is computed with different settings of boundary voltages. In this case, A, which is determined solely by conductor shape, does not change, and b, which is determined by boundary voltages, does.

By saving the triangularization results from the first Gauss elimination, subsequent solutions for

---

[10]An addition, subtraction, multiplication, or division is classified as one operation. While these execute with somewhat different speeds, the ratios of each operation type remain fairly constant within each operation step.
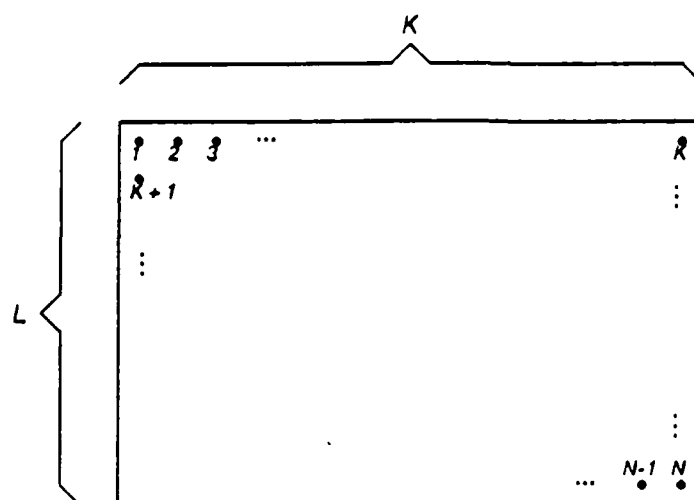
Figure B-1: Rectangular region divided into two-dimensional node grid

(a)

(b)
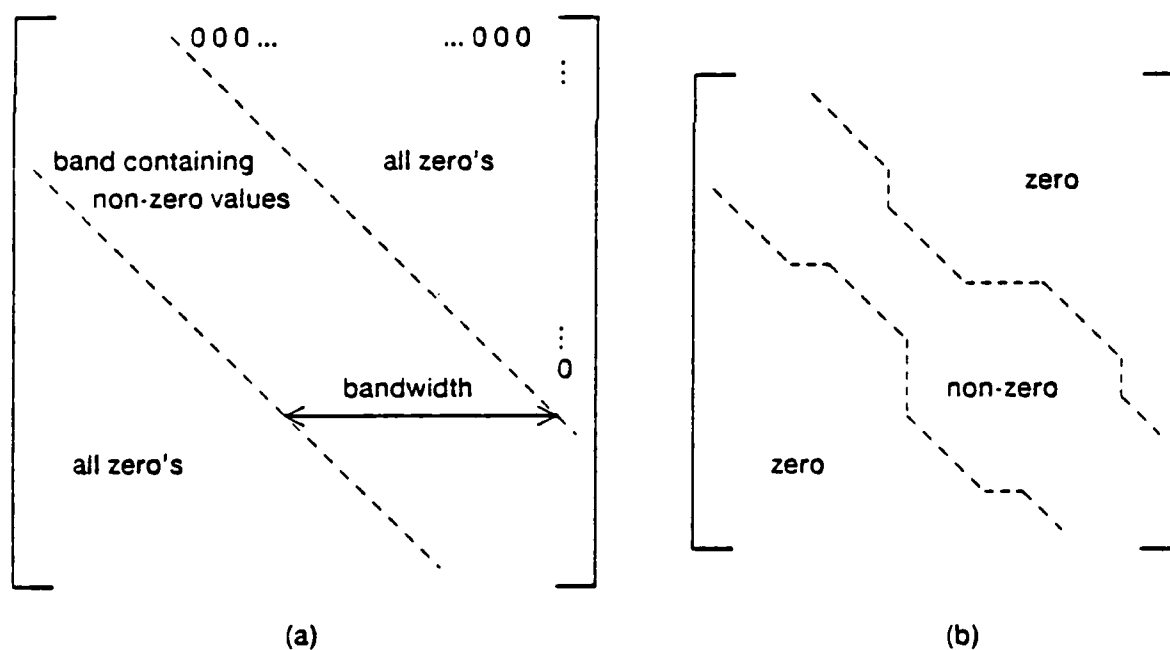
Figure B-2: Band matrices

(a) regular (b) irregular.

APPENDIX B

Band-Matrix Operations

Algorithms for band-matrix operations have been well developed, because of their commonplace usage for finite element problems. Band-matrices have a known structure that enables the special algorithms to save on storage requirements and computation requirements [15].

To visualize how band-matrices arise, consider the two dimensional region shown in figure B-1. The region is divided into a two-dimensional node grid with L rows and K nodes per row. The nodes are numbered sequentially from 1 to N in a horizontal raster method. The system of equations that results from describing each point by its four nearest-neighbors has a band-matrix shape like that shown in figure B-2(a). All non-zero matrix entries are located inside a band centered on the matrix diagonal. The overall matrix dimension is $N \times N$, where $N = KL$. The *bandwidth*, M, is determined by the grid size: $M = 2K + 1$ for a four-point nearest-neighbor system, or $M = 2K + 3$ for a nine-point nearest-neighbor system. If the finite element region is irregularly shaped, then the system matrix is an *irregular band-matrix*. However, the irregular system can use regular band-matrix procedures where the bandwidth is determined by the longest horizontal row.

B.1 Band-Matrix Storage

The savings in band-matrix storage comes by not storing the zero values outside of the band. These zero values are never needed. For each row, the starting index is stored, followed by the M values for that row. This reduces the storage requirement from $N \times N$ to less than $N \times M$. For a square node region, $M = (2 \operatorname{Log}_2 N) + 1$, and the storage needs are approximately $2N \operatorname{Log}_2 N$.

116

Number of stages	Relative Error (in percent)
1	18.9
2	5.0
3	2.3
4	1.3

Table A·1: Relative error of minimum pole for n-stage π-ladder

The actual error of delay time is given by

$$\Delta t_{0.9} = RC \cdot \text{REMP}(n).$$

By specifying a maximum $\Delta t_{0.9}$ the number of stages can be determined for any conductor with a known R and C. In practice, only long straight conductors must be divided beyond that of internal subregion divisions.

As stated earlier, including a non-zero drive resistance and/or a non-zero load capacitance, the REMP of the π-ladder approximation can be improved substantially. For instance, in a three-stage π-ladder network with $R_T = R$ and $C_T = 0$, the REMP reduces from 2.3% to 0.3%. Unfortunately, the values of R_T and C_T are unknown to the extractor when the π-ladder network is developed. The error values given here are worst case values. Under many cases, the real modelling errors are much less due to non-zero R_T's, non-zero C_T's, and the node breaks which are always present at connection branches.

the π-ladder accuracy since the load capacitance and drive resistance are exact components, and the effects of these elements become more prominent as R_T and C_T increase. From Heaviside's expansion theorem, we can express the step-response voltage at the load end, $v_2(t)$, as

$$\frac{v_2(t)}{V_{cc}} = 1 + \sum_{k=1}^{\infty} C_k e^{-\sigma_k \frac{t}{RC}}. \tag{A.1}$$

Peirson [26] has evaluated the poles, σ_k, and coefficients, C_k, for the distributed case:

$$\sigma_k = \left(k - \frac{1}{2}\right)^2 \pi^2, \text{ and} \tag{A.2}$$

$$C_k = \frac{2(-1)^k}{\pi\left(k - \frac{1}{2}\right)}.$$

For an n-stage π-ladder network,

$$\sigma_k(n) = 4\left[n \sin\left(\frac{2k-1}{4n}\,\pi\right)\right]^2, \tag{A.3}$$

and the C_k coefficients are computed numerically as described by Sakurai. In both of these cases, the minimum pole, σ_1, is much less than the next pole, σ_2. Thus, equation (A.1) can be approximated by

$$\frac{v_2(t)}{V_{cc}} = 1 + C_1 e^{-\sigma_1 \frac{t}{RC}}, \tag{A.4}$$

and σ_1 can be considered the time constant. The approximation effects the waveform only at the start of the output voltage transition, but is very close to the true solution when the voltage nears 90% of its transition. Propagation time is given by

$$t_{0.9} = \frac{RC}{\sigma_1} \ln(10 C_1).$$

From this we see that delay is inversely proportional to RC, and the relative error of delay, $\Delta t_{0.9}$, is proportional to the relative error of the minimum pole (REMP). The REMP for the n-stage π-ladder network is

$$\text{REMP}(n) = \frac{[\sigma_1(n) \text{ of ladder circuit}]}{[\sigma_1 \text{ of distributed } RC]} - 1$$

and has been calculated from equations (A.2) and (A.3). Table A-1 lists REMP values for π-ladder circuits with one through four stages. With a three stage ladder network, we see that the relative delay error is always less than 2.3%, which is usually quite sufficient for IC extraction models.

APPENDIX A

Distributed *RC* Modelling with π-Ladders

Integrated circuit conductors tend to behave as distributed *RC* lines. In EXCL, these are modelled with discrete elements in an *n*-stage π-ladder topology, as shown in figure A-1. In this section, we will analyze the error that is present in the π-ladder equivalent of a distributed *RC*.

(a)

(b)
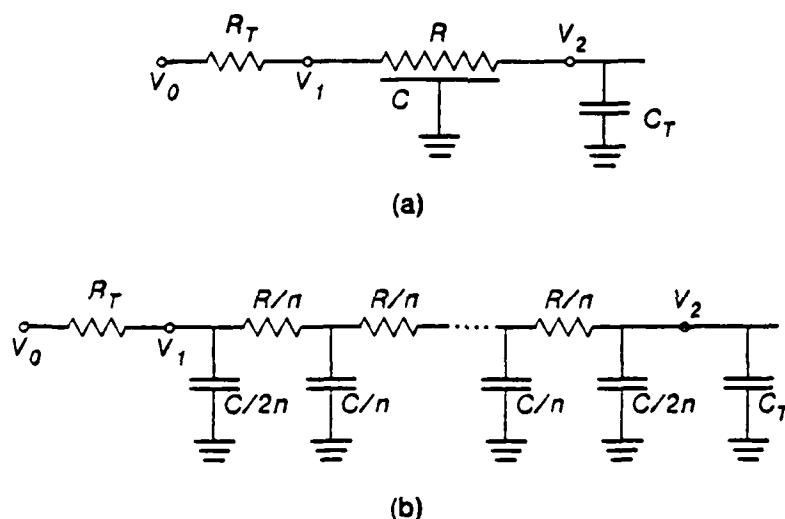
Figure A-1: π-ladder network

(a) Distributed *RC* interconnection, (b) equivalent π-ladder network

Our figure of merit will be $\Delta t_{0.9} = |t_{dist,0.9} - t_{\pi,0.9}|$, the difference in step-response delay time between the distributed *RC* line and the π-ladder network. The voltage v_0 is a step from zero to v_{cc} and the subscript, 0.9, indicates that delay times measure when v_2 reaches $0.9v_{cc}$.

First, we assume that both R_T and C_T are zero. Non-zero values for either only tend to improve

7.1 Final Notes and Recommendations for Improvement

For each extraction problem, the solutions for straight and library subregions represent the best possible, for they are fast and accurate. Any time that extraction reverts to a general numerical solution the execution time soars. A carefully selected library enhances the extractor, and for the best performance, the library should contain as many potential regions as possible.

A brief study of the irregular regions in some test layouts has shown that a large portion of the shapes occur more than once. This should seem obvious, considering that IC design is largely hierarchical, and that a few basic cells are instantiated many times. However, often within a single cell one finds equal irregular regions. If the extractor includes a *dynamic library*—that is, a library capable of accepting new entries at any time—then each "general" solution to an irregular region can be added. Any repetitions of the region automatically find their solution in the library.

Dynamic libraries should be considered for a partial alternative to hierarchical extractors, since repetitious layouts benefit most from dynamic libraries. If we approximate extraction time as the time needed for just the general numerical solution, then essentially, a cell is extracted once regardless of how often it is instantiated. With this technique, it is not left up to the integrity of the hierarchy to guarantee the absence of careless overlaps, for the extractor will always pick out fluctuations in individual cell instances.

In chapter 5 we saw that initializing the general and parallel inter-nodal capacitance methods for a technology depends on a large sequence of computer field simulations. Presently, each simulation requires manual set-up. A vast improvement is feasible with an automatic capacitance calibrator, a program that generates all capacitance information from a simple geometric description of the process. An automatic capacitance calibration system would enable one to make more thorough checks of the inter-nodal capacitance calculations than is now possible.

Finally, manhattan geometries prevail in most IC designs, and in fact some of the most complex IC's to date have been satisfactorily designed with only orthogonal rectangles . However, some layout wizards insist that 45° rectangles give an essential extra flexibility. Currently, EXCL extracts all regions containing 45° angles with general methods, but it may be possible to improve this. It may be feasible, for instance, to isolate the long straight regions in 45° strips by "subdividing" the rectangles twice—once on the up and down rectangles and once on the 45° rectangles. By including a new mask object type, the 45° box, data representation could be greatly compressed, and library entries could be added more easily.

true when it counts the most—when calculating the capacitance between two conductors with large amounts of coupling between opposing edge faces.

The best way to keep the extraction problems from growing to unmanageable complexity is to use simpler computation methods where possible. This is, fortunately, possible at many locations on an IC where field conditions are known. An automatic subdivision algorithm isolates long, straight regions and commonly-occurring library regions with precomputed answers. For these regions it is possible to extract a parameter rapidly. The savings from using the rapid techniques where possible over using the general technique everywhere clearly depends on the regions's shape, but for typical layouts the extractor operates an order of magnitude faster. Most significant, is that the improvement in extraction time is possible with only minute losses of accuracy. The same basic algorithm (presented in chapter 4) functions for each of the three time consuming extractions: resistance, inter-nodal capacitance, and transistor sizing.

The extractor remains flexible through its highly modular organization. A user needs to look at just two program modules to make fabrication process or extraction model changes. The user can pattern EXCL to accept a new fabrication process by changing only the module containing all mask intersection rules. In one instance, an (experienced) user altered a CMOS extractor into an NMOS extractor in about thirty minutes. Unlike most other extractors, the user of EXCL can also change the circuit generation rules. He can, for instance, change a detailed circuit extractor for SPICE simulations into a more rapid switch level extractor.

One final observation is evident through many of the chapters. Wherever possible, EXCL eliminates insignificant circuit data or insignificant parameter extractions. Minimum circuit values and extraction tolerances can be set by the user, thus providing him with an "accuracy knob". On a first pass through the extractor with any layout, the user sets the knob low. The answers will be less accurate, but extraction is fast and will hopefully provide enough information to make the first analysis. Once the user has a more refined layout, he can turn the knob up and extract a more accurate circuit characterization.

111

CHAPTER SEVEN

Conclusions

The extraction methods and algorithms presented in this thesis were developed to fill a need for automated and accurate circuit characterization of IC layouts. An outcome of the project is an automatic circuit extractor that, when combined with a circuit simulator or other analysis tool, permits an IC designer to detect and correct circuit flaws without ever having the design fabricated. The following guidelines directed many of the decisions during development of the circuit extractor.

1. The extractor should generate an equivalent circuit that provides all possible accuracy in predicting the true behavior of the integrated circuit. Two consequences follow from this. First, extraction must find *all* relevant circuit parameters, including the difficult ones— resistance and coupling capacitance. Second, the extractor should compute each circuit parameter with as little error as reasonably possible.

2. *Extraction time should stay at a minimum. The appeal of circuit extraction diminishes with long waits and vast computations.*

3. One should be able to modify the extractor with some ease for different types of subsequent analysis.

4. The extractor should be amenable to new fabrication processes or mask specifications.

After inspecting a sampling of IC layouts, it was decided that extraction should not make only the easy calculations, i.e., the calculations of resistance along straight conductors or capacitance between parallel conductors. It must also calculate values around irregularly shaped conductors.

General numerical methods were adopted for calculating circuit parameters around the irregular areas. While the resistance and to some extend the transistor sizing problems each have known general methods, no adequate method was found for the problem of coupling capacitance extraction between irregular conductors. One was developed as described in chapter 5. Its calibration to a particular process is based on computer simulations of measured cross-sectional conductor shapes. The method's performance surpasses that of other general capacitance extractors. This is especially
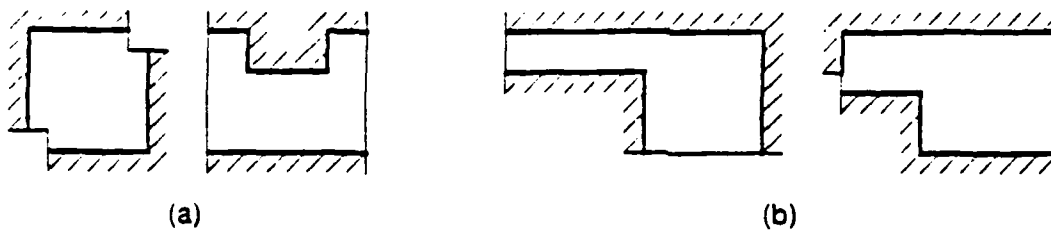
(a)　　　　　　　　　　　　　(b)

Figure 6-3: Samples of transistors that cannot be sized

6.2.3 Irregular Subregion Dimensions

No true method exists for computing the conduction dimensions in an irregular subregion, for the current lines have different lengths. For such regions the extractor assumes one of the dimensions: the length for a short-and-wide transistor or width for a long-and-narrow transistor. This dimension should be known based on values from the other subregions. ExcL computes the other dimension from the expression:

$$R_{eq} = \frac{Length}{Width} ,$$

where R_{eq} is the equivalent resistance of a similar shape with a constant, unit sheet resistance. The equivalent sheet resistance is computed with the general methods discussed in the resistance extraction chapter (chapter 4).

Transistors which are so contorted (figure 6-3(a)) that they contain only one irregular subregion are not sized. An error message is given to the user, since these usually indicate a layout error or a special-case layout that is better characterized by the designer.

6.2.4 Dimension Combination and Coherency Check

The length and width information of all subregions must be compiled into a single value for both length and width. First, a coherency check is made for the length of each subregion in a short-and-wide transistor or the width of each subregion in a long-and-narrow transistor. From the last section we see that technically, only straight and library subregions are checked. All these values should be equal; otherwise, an error message denotes the discrepancy. The other non-equal dimension (width for short-and-wide transistors or length for long-and-narrow transistors) is computed by summing each subregion's dimension. Refer to figure 6-1 for samples of computed transistor shapes. Figure 6-3(b) shows transistor shapes failing the coherency check.

6.2.1 Straight Subregion Dimensions

The transistor length and width of a straight subregion are simply the two dimensions of the subregion's only rectangle. The extractor matches transistor "length" and "width" with the rectangle dimensions by examing the connections to the rectangle. For a short-and-wide transistor, the transistor length is measured between *node 1* and *node 2* connections, while transistor width is measured along either node connection. For a long-and-narrow transistor, length is measured between the only two connections (internal or nodal), while transistor width is measured along a connection.

6.2.2 Library Subregion Dimensions

Even though connections are ignored when dividing the region into subregions, they must be considered when looking up entries in the library. All fetches from the transistor dimension library return explicit values for both length and width. These values are stored in the library, but, scale independence of <u>shapes</u> is maintained by normalizing all library values to the dimension η defined by the length of the topmost rectangle edge. The true dimensions are calculated by substituting η with the true topmost rectangle dimension. Figure 6-2 shows some interesting library entries. The first two form at right angle corners of both types of transistor; the other is frequently found at butting contact connections of NMCS pullup transistors.
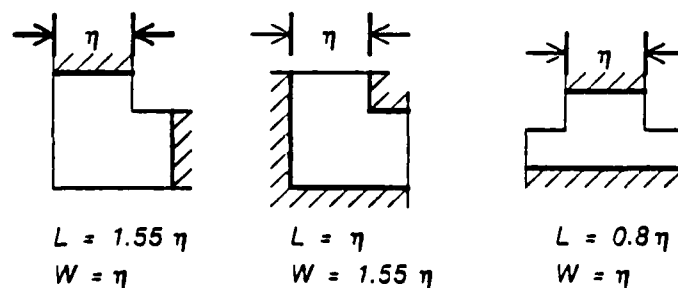

$L = 1.55\,\eta \qquad\qquad L = \eta \qquad\qquad L = 0.8\,\eta$

$W = \eta \qquad\qquad W = 1.55\,\eta \qquad\qquad W = \eta$

Figure 6-2: Entries of transistor dimension library

107

6.2 Non-Rectangular Transistors

With the remaining non-rectangular transistors, the dimension extraction proceeds along much the same track as resistance extraction. First, however, the extractor decides whether the transistor is "short-and-wide" or "long-and-narrow". This decision is based on how many of the non-intersecting channel rectangles touch both *node 1* and *node 2* connections. A rectangle count greater than or equal to two signifies a short-and-wide transistor, while a number less than two signifies a long-and-narrow transistor.
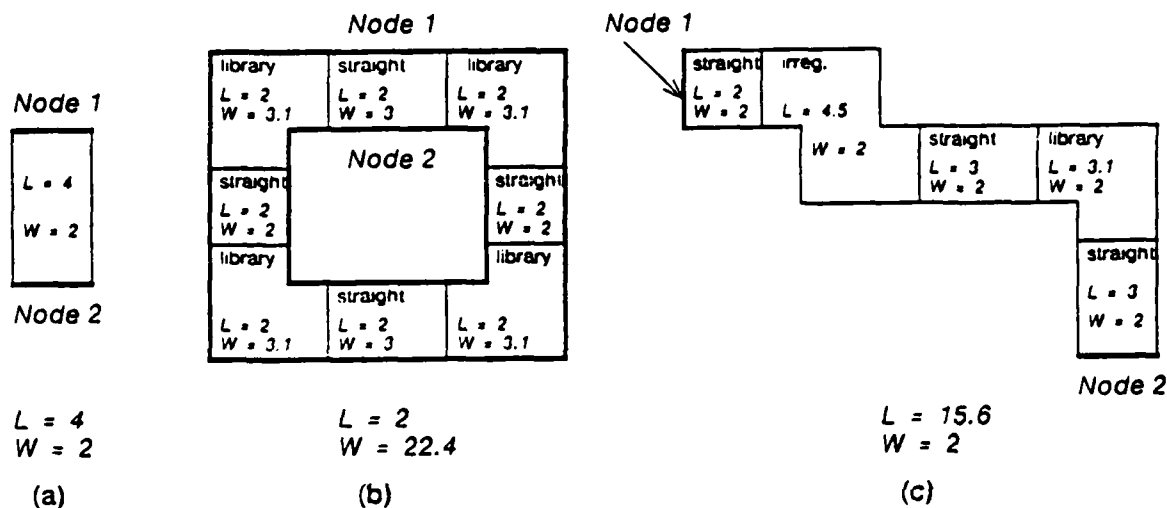

Figure 6-1: Samples of transistors with calculable dimensions

A rectangular transistor (a) has only one rectangle. Length and width are given directly by the rectangle's dimensions. A short-and-wide transistor (b) or long-and-narrow transistor (c) is subdivided, and the dimensions of each subdivision are found. Afterwards, the separate dimensions are compared and combined.

Next, the extractor divides the channel area into subregions as though it were a resistor region with no connections. That is, the *node 1* and *node 2* connections are ignored during subdivision. The subdivision technique is described in section 4.5. Figures 6-1(b) and 6-1(c) show samples of the subregions for a short-and-wide and long-and-narrow transistor. The extractor computes the length and width for each subregion by following the techniques outlined in the following sections. Afterwards, all the separate subregions' dimensions are combined and checked for coherency.

106

APPENDIX C

Spherical Coordinate Simulations

Since the effects of a unit elemental charge posses spherical symmetry on an IC substrate, a spherical simulator was developed to determine the weighted-Green's functions described in chapter 5. Here we examine the significant modifications to the basic Laplace (Poisson) Equation simulator which were necessary for spherical coordinate simulation.

Figure C-1: Setup of finite element analysis for determination of Greens's functions.

The area around the point charge is divided into finite element rings as shown in figure C-1. Each ring has constant voltage, since there is no variation of voltage with θ. Poisson's equation in spherical coordinates with $\dfrac{\partial V}{\partial \theta}$ set to zero is [27]

$$-\frac{\rho}{\epsilon} = \nabla^2 V = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial V}{\partial r}\right) + \frac{\partial^2 V}{\partial z^2}.$$

An approximate difference equation for the finite element, o, and its four nearest neighbors is

$$-\frac{q}{\epsilon} = \frac{1}{r_i}\left[\frac{r_i + r_{i-1}}{2}(V_{r-} - V_o) - \frac{r_i + r_{i+1}}{2}(V_o - V_{r+})\right] + (V_{z-} - V_o) - (V_o - V_{z+}).$$

Since the distance between all points is a,

$$-\frac{q}{\epsilon} = \left(1 - \frac{a}{2r_i}\right)(V_{r-} - V_o) - \left(1 + \frac{a}{2r_i}\right)(V_o - V_{r+}) + (V_{z-} - V_o) - (V_o - V_{z+}),$$

or

$$V_o = \frac{1}{4}\left[\left(1 - \frac{a}{2r_i}\right)V_{r-} + \left(1 + \frac{a}{2r_i}\right)V_{r+} + V_{z-} + V_{z+} + \frac{q}{\epsilon}\right].$$

The spherical field simulator uses this equation for the basic operation at each point. Other than the inclusion of the two correction factors for V_{r-} and V_{r+}, the simulator operates identically to the rectangular simulator. It uses either the direct, Gauss elimination method or iterative method of solution.

References

[1]   B. T. Preas, B. W. Lindsay and C. W. Gwyn, "Automatic Circuit Analysis Based on Mask Information," *Proceedings of the 13th Design Automation Conference*, San Francisco, June 1976, pp. 309-317.

[2]   C. Baker, "Artwork Analysis Tools for VLSI Circuits," Master's thesis, Massachusetts Institute of Technology, June 1980.

[3]   T. Mitsuhashi, T. Chiba and M. Takashima, "An Integrated Mask Artwork Analysis System," *Proceedings of the 17th Design Automation Conference*, Minneapolis, June 1980, pp. 277-284.

[4]   A. E. Ruehli, "Survey of Computer-Aided Electrical Analysis of Integrated Circuit Interconnections," *IBM Journal of Research and Development*, Vol. 23, No. 6, November 1979, pp. 626-639.

[5]   J. D. Bastian, et. al., "Symbolic Parasitic Extractor for Circuit Simulaion (SPECS)," *Proceedings of the 20th Design Automation Conference*, San Diego, June 1983, pp. 346-352.

[6]   C. A. Mead, L. Conway, *Introduction to VLSI Systems*, Addison Wesley, Reading, Mass., Addison-Wesley Series in Computer Science, 1980.

[7]   R. E. Bryant, *A Switch-Level Simulation Model for Integrated Logic Circuits*, PhD dissertation, Massachusetts Institute of Technology, March 1981.

[8]   C. J. Terman, *Simulation Tools for Digital LSI Design*, PhD dissertation, Massachusetts Institute of Technology, August 1983.

[9]   A. Vladimirescu, Kaihe Zhang, A. R. Newton, D. O. Pederson, and A. Sangiovanni-Vincentelli, "SPICE Version 2G.5 User's Guide," Tech. report, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, August 1979.

[10]   A. Vladimirescu, "The Simulation of MOS Integrated Circuits Using SPICE2," Tech. report, Department of Electrical Engineering and Computer Science, University of California, Berkeley, June 1979.

[11]   P. Penfield Jr., "Signal Delay in RC Tree Networks," *Proceedings of the 18th Design Automation Conference*, June 1981, pp. 613.

[12]   Takayasu Sakurai, "Approximation of Wiring Delay in MOSFET LSI," *IEEE Journal of Solid State Circuits*, Vol. SC-18, No. 4, August 1983, pp. 418-426.

[13]   J. L. Bentley, D. Haken and R. W. Hon, "Fast Geometric Algorithms for VLSI Tasks," *Digest of Papers, 20th IEEE Computer Society International Conference (CompCon),* , San Francisco CA, February 1980, pp. 88-92.

[14]   J. L. Bentley, D. Haken, and R. Hon, "Statistics on VLSI Designs," Technical Report CMU-CS-80-111, Department of Computer Science, Carnegie-Mellon University, April 1980.

[15]   J. R. Whiteman, *The Mathematics of Finite Elements and Applications*, Academic Press, London, 1973, pp. 427-447.

[16]   J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

[17]   W. E. Milne, *Numerical Solutions of Differential Equations*, Wiley, New York, 1953, pp. 216-217.

[18]   G. E. Forsythe and W. R. Wasow, *Finite-Difference Methods for Partial Differential Equations*, Wiley, New York, 1960.

[19]   P. M. Hall, "Resistance Calculations for Thin Film Patterns," *Thin Solid Films*, Vol. 1, 1967/68, pp. 277-295.

[20]   Philip Balaban, "Calculation of the Capacitance Coefficients of Planar Conductors on a Dielectric Surface," *IEEE Transactions on Circuit Theory*, Vol. CT-20, No. 6, November 1973, pp. 725-731.

[21]   Peter Benedek, "Capacitance of a Planar Multiconductor Configuration on a Dielectric Substrate by a Mixed Order Finite-Element Method," *IEEE Transactions on Circuits and Systems*, Vol. CAS-23, No. 5, May 1976, pp. 279-284.

[22]   P. D. Patel, "Calculation of Capacitance Coefficients for a System of Irregular Finite Conductors in a Dielectric Sheet," *IEEE Transactions on Microwave Theory and Techniques*, Vol. MTT-19, No. 11, November 1971, pp. 862-869.

[23]   R. L. M. Dang, and N. Shigyo, "Coupling Capacitances for Two-Dimensional Wires," *IEEE Electron Device Letters*, Vol. EDL-2, No. 8, August 1981, pp. 196-197.

[24]   P. Silvester, "TEM Wave Properties of Microwave Transmission Lines," *Proc. Inst. Elec. Eng.*, Vol. 115, No. 1, January 1968, pp. 43-48.

[25]   W. H. Dierking and J. D. Bastian, "VLSI Parasitic Capacitance Determination by Flux Tubes," *IEEE Circuits and Systems Magazine*, Vol. 4, No. 1, March 1982, pp. 11-18.

[26]   R. C. Peirson and E. C. Bertnolli, "Time-domain analysis and measurement technique for distributed RC structures.," *Journal of Applied Physics*, Vol. 40, No. 1, January 1969, pp. 118.

[27]   William H. Hayt, *Engineering Electro-Magnetics*, McGraw-Hill, New York, 1974.

END

FILMED

5-85

DTIC